Real-time SLAM for Humanoid Robot Navigation Using Augmented Reality

by

Yixuan Zhang

B.Sc., Shenyang Jianzhu University, 2010

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Applied Science

in the
School of Mechatronic Systems Engineering
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY

Spring 2014

Approval

Name: **Yixuan Zhang**

Degree: **Master of Applied Science**

Title of Thesis: ***Real-time SLAM for Humanoid Robot Navigation Using Augmented Reality***

Examining Committee: **Chair:** Siamak Arzanpour
Associate Professor

Ahmad B. Rad
Senior Supervisor
Professor

Gary Wang
Supervisor
Professor

Carlo Menon
Internal Examiner
Associate Professor
School of Engineering Science

Date Defended/Approved: April 03, 2014

Partial Copyright Licence

Abstract

The integration of Augmented Reality (AR) with Extended Kalman Filter based Simultaneously Localization and Mapping (EKF-SLAM) is proposed and implemented on a humanoid robot in this thesis. The goal has been to improve the performance of EKF-SLAM in terms of reducing the computational effort, to simplify the data association problem and to improve the trajectory control of the algorithm. Two applications of Augmented Reality are developed. In the first application, during a standard EKF-SLAM process, the humanoid robot recognizes specific and predefined graphical markers through its camera and obtains landmark information and navigation instruction using Augmented Reality. In the second stage, iPhone on-board gyroscope sensor is applied to achieve an enhanced positioning system, which is then used in conjunction of a PI motion controller for trajectory control. The proposed applications are implemented and verified in real-time on the humanoid robot NAO.

Keywords: Augmented Reality; EKF-SLAM; Humanoid robot NAO; iPhone gyroscope

To my parents and all the other people I love

Acknowledgements

Foremost, I would like to express my sincere gratitude to my senior supervisor Dr. Ahmad Rad, whose patience, understanding, enthusiasm, and immense knowledge added considerably to my graduate experience. During these years of study, I have encountered a lot of challenges; fortunately he is a great coach to encourage me and try his best to help me to overcome those difficulties. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Masters study. I would like to thank him once again from the deep of my heart.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Gary Wang, Dr. Siamak Arzanpour and Dr. Carlo Menon for their encouragement, insightful comments, and hard questions. I am deeply grateful to them for being on my committee and reviewing my thesis and gave me lots of valuable advices about my thesis.

I thank my fellow labmates in Autonomous and Intelligent Systems Laboratory (AISL): Mohammad Al-Qaderi, Mehdi Cina, Mehran Shirazi, and Kamal Othman for the enjoyable collaboration and interesting discussions we had.

Last but not the least, I would like to thank my parents: Jie Zhang and Fei Sun, for giving birth to me at the first place and supporting me spiritually throughout my life.

Table of Contents

List of Tables

List of Figures

Chapter 1.

Introduction

At this particular time of human's history, it is envisaged that the current technology allows us to design "human-like" machines. Humanoid robots are the first generation of such systems and have attracted significant research and public interest over the last two decades. The emergence of cost effective robots such as NAO [1] in recent years have facilitated research in many areas from sensing and perception, obstacle detection, dynamic stability, gait generation ,real-time control, navigation and path planning to social robotics [2]. In particular, many studies have addressed the problem of humanoid robot navigation through an unknown environment using on-board sensors such as laser and vision [3-6]. In order to provide a robot with navigation capabilities, it should be able to obtain a working map of the environment as well as its own position within the map. However, this information is not generally available when the environment is not known a priori (i.e. indoor) and the robot cannot position itself in that environment. Considering the example of indoor spaces (I-space): different from outdoor spaces (O-space), I-space is generally in a smaller scale than O-space, and O-space positioning technologies are not applicable to I-space such as GPS and other technologies are used such as Wi-Fi and RFID [7]. Whereas, typical indoor environments are generally partially known (i.e. recognizable landmarks) and one can use the known information to improve the navigation process. Research in the last two decades has studied the above problems either in isolation or together towards the realization of autonomous robots. Such robots have the capability to solve the problem of Simultaneous Localization and mapping (SLAM). While the SLAM problem has been intensively researched, many challenges still remain, such as map representation, data association, localization, and computational complexity [8]; the problem is still open for further research and is studied in this thesis. To provide contributions to the SLAM problem, this thesis proposes a novel approach that integrates probabilistic based

landmark SLAM algorithm with a popular vision based technology Augmented Reality (AR).

Augmented Reality is a technology that brings an enhancement to a human or a machine perception of an environment by combining computer-generated sensory input to the view of physical environment. Specifically, it has been recently broadly used for indoor and outdoor navigation in multiple platforms such as mobile phone and head-mounted display (HMD) for the convenience of human, but very few implementations are found in robotics. The primary objective of this thesis is to implement AR technology on a humanoid robot NAO in order to enhance and complement Extended Kalman Filter (EKF) based probabilistic SLAM algorithm in an indoor environment, which has previously been employed onto the NAO robot in thesis [9]. Augmented Reality is applied in this thesis intending to provide original EKF-SLAM algorithm with improvement on computational demand, simplification on the data association process, and enhancement on the trajectory path of the robot. Research proposed in this thesis is a continued work in the Autonomous and Intelligent Systems Laboratory (AISL) based on [10-13].

1.1. Objectives

The rationale for this project is to address supplementary solutions to standard SLAM (Simultaneous Localization and mapping) problem. We argue that indoor environments are partially known and other technologies could enhance the performance of standard SLAM algorithm. In particular, we employ augmented reality as an additional feature that could improve the SLAM solution. Most current research on augmented reality addresses its application for overlaying virtual information on real information for human use. In contrast, we argue that a robot could also benefit from augmented reality by using additional instruments to “augmented its understanding of the environment”.

In this context, the preliminary objectives in this project are summarized as follow:

- Encoding AR-EKF-SLAM algorithm in Python programming language (Chapter 4,5)
- Implementing full AR-EKF-SLAM on humanoid robot NAO (Chapter 5)
- Experimentation and validation on AR-EKF-SLAM and result analysis and discussion (Chapter 5,6)

1.2. Thesis outline

This thesis presents the implementation of Augmented Reality onto the SLAM problem of a humanoid robot NAO. The presentation is organized in several chapters that discuss the theoretical and experimental approaches of the proposed implementation.

Chapter 2 provides a selected and directed literature review on three topics related to this study. A survey on different types of mobile robots with a focus on biped robots is presented first. Then the technology of Augmented Reality and its applications are reviewed. Solving robot localization and mapping problem by using probabilistic approach in SLAM is introduced at last.

Chapter 3 presents a comprehensive overview of the robot platform NAO humanoid robot used in this thesis. NAO's specification, software framework, and implementation methods are included.

In chapter 4, EKF-SLAM is demonstrated including a comprehensive description of the algorithm and its simulation and real-time implementation. EKF-SLAM is interpreted with the description of SLAM components and SLAM process, and in the implementation section, simulated and real experimental results with different number of landmarks are also demonstrated.

Extensive studies based on EKF-SLAM and Augmented Reality integration are discussed in Chapter 5. This chapter includes two distinctive applications of Augmented Reality integrated with the original EKF-SLAM program, which is considered as the main contribution of this thesis.

In Chapter 6, the results from the original EKF-SLAM experiment and from the two applications of Augmented Reality are presented and compared in order to clarify the improvement of AR-EKF-SLAM algorithm.

In Chapter 7, conclusions, contributions and future works are discussed.

Chapter 2.

Literature review

Conducing a thorough literature review prior to starting my research project was instrumental in my understanding of the background theory and provided me with valuable information regarding the state-of-the-art of the technology and related methodologies adapted by other researchers around the world . This chapter mainly consists of three parts: a summary of literature survey on different types of mobile robots with a focus on land-based robots, continued with solving the robot localization and mapping problem using probabilistic approach in SLAM, and followed by an overview of Augmented Reality with its applications in robotics. Figure 2.1 demonstrates the related areas and the scope of the thesis. Those shown in highlighted squares designate the areas that are directly addressed in the thesis. Bold entries are covered in this thesis.

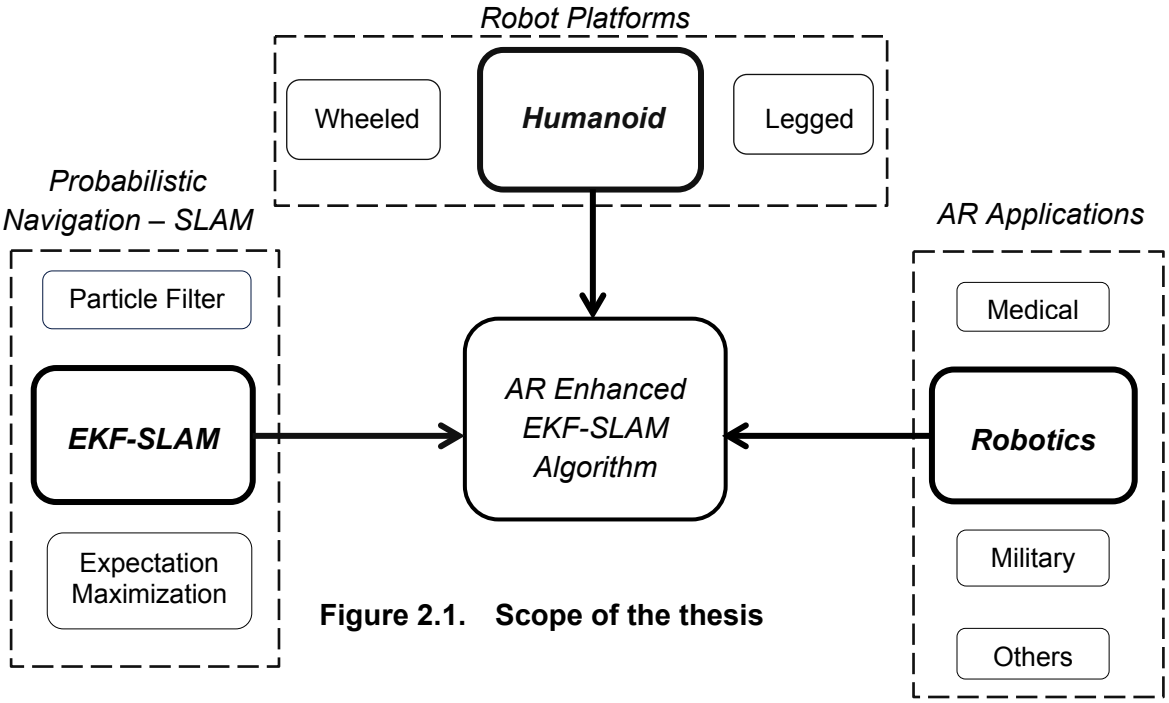


Figure 2.1. Scope of the thesis

2.1. Mobile robots

A Mobile Robot is common platform for robotic navigation studies and is capable of navigating through an environment with limitless operational area. It has many different applications stretching from home entertainment for children, through rescue and secure missions as well as military assistance, to universe exploration [14]. In order to complete the task, various types of robots has been designed, which are mainly classified by the environment they operate within. These include

Land-based robots, usually called *Unmanned Ground Vehicles* (UGVs) [15].This type of robot travels on the surface of ground either with no human presence or carry passengers onboard. A variety of applications can be found for this type in the field of civil transportation, material handling, military assistance, healthcare for elderly and the disabled, security and entertainment.

Air-based robots, often named *Unmanned Aerial Vehicles* (UAVs) [16]. This type of robot operates in the air and has no human pilot on board. Applications usually found in autonomous planes, helicopters, and Blimps.

Water-based robots also referred to as *autonomous underwater vehicles* (AUVs) [17]. The robot is able to travel underwater without the manipulation from human. In operation, AUVs is remotely controlled by an operator from the surface and the most common application areas for AUVs are military operation and undersea scientific research.

The SLAM problem is common among all the above categories of autonomous robots. Among the three general types of robot above, land-based robots are the most popular in either universities academic research or real applications. However, the applications of a land robot can be distinguished by the fundamental designs within this type are limited to three: tracked, wheeled and legged (or bi-biped). Following sections introduce two of the most common approaches of locomotion for land-based robots: using wheels and legs. The respective advantages and disadvantages along with the suitable situations for selected types are discussed.

2.1.1. Wheeled Mobile Robot

Wheeled robots are robots that travel on the ground via several motorized wheels. For navigation on the most common type of ground surfaces such as flat and less rough terrain, the design of a mobile robot using wheels is relatively simpler than the tread or legged robot. Therefore, the wheeled robot is the most popular solution among other types of robot locomotion and has been used to propel robots and robotic platforms of different sizes. There is a wealth of different technical designs for wheeled robots, which can be generally differentiated by the number of wheels equipped.

- Two-wheeled robots

  A two wheeled robot is also referred to dicycles, mounted with two motorized wheels at left and right side of robot. For this type of robot, it is easy to imagine that the main issue is to keep upper body upright at movement. In order to do this, the two wheels must keep moving in the direction that the robot is failing. Thus, a common design of two wheel robot with good balance usually has its batteries underneath the body to ensure a low center of gravity of the robot [18]. For example, Nbot (figure 2.2) uses inertial sensor and position information from encoder to balance [19].

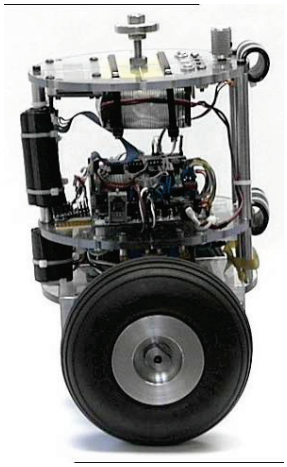


Figure 2.2. Two-wheeled balancing robot Nbot [19]

- Three-wheeled robots

A three-wheeled robot usually propels by two powered wheels plus a free turning wheel. In figure 2.3 we can see that three wheels on the robot are installed in a triangle to balance. The center of balance of the robot is recommended to be designed as close to the center of the triangle as possible to keep stable while driving. To change directions, the relative rotation of each powered wheel is commanded at different rate based on the amount of turning required. For example, the robot will go along a straight line if both wheels rotate at a same speed. This type of driving system is commonly called differential steering system [20]. Two pioneer three wheeled-robots with differential steering wheels are used as project platforms in my AISL research lab (Figure2.4).

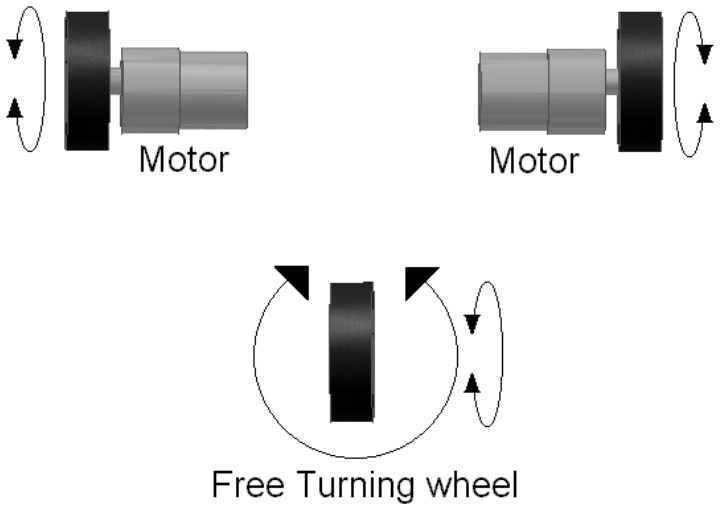


Figure 2.3. Three-wheeled robot using differentially steered system [14]

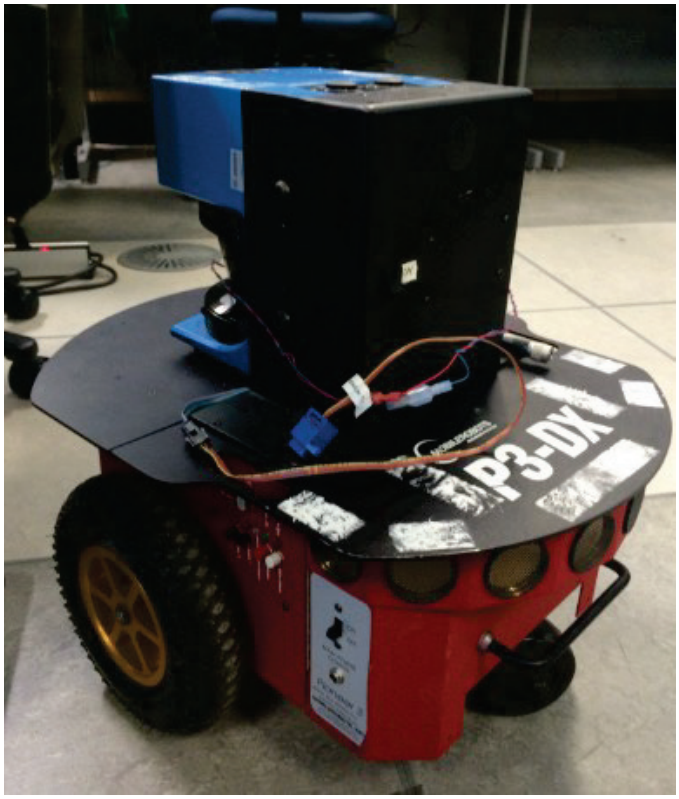


Figure 2.4. Three-wheeled pioneer robot in AISL lab

2.1.2. Legged Robot

Legged robots achieve mobility using several mechanical legs. Legged locomotion robot if designed properly have better mobility than wheel locomotion on very uneven, rough terrain due to the irregularity of ground condition. Alike wheeled robots, the number of legs for a legged robot can be different, whereas each leg must have at least two degrees of freedom (DOF) to make it mobile. For each DOF one joint is a required, which is commonly powered by one servo [21].In the following, select most popular leg configurations are shown and discussed.

- Two-legged robots/Humanoid Robots

  Two legged or bi-pedal robots move in the same way that human does. The studies on biped robots have become one of the most popular topics in the last decade for the obvious reason of similarity of locomotion between the biped robot and humans. Due to the nature of biped robots, most of the research has been

focused on humanoid robots, an autonomous robot with human form and human-like abilities [22]. A well-known application of bi-pedal humanoid robot is NAO robot (Figure2.5), developed by Aldebaran Robotics in France. The robot is able to do various tasks that a human does, such as walk, stand, pick and place, and even dance. NAO is also selected to be my research platform and will be introduced in details in next chapter.

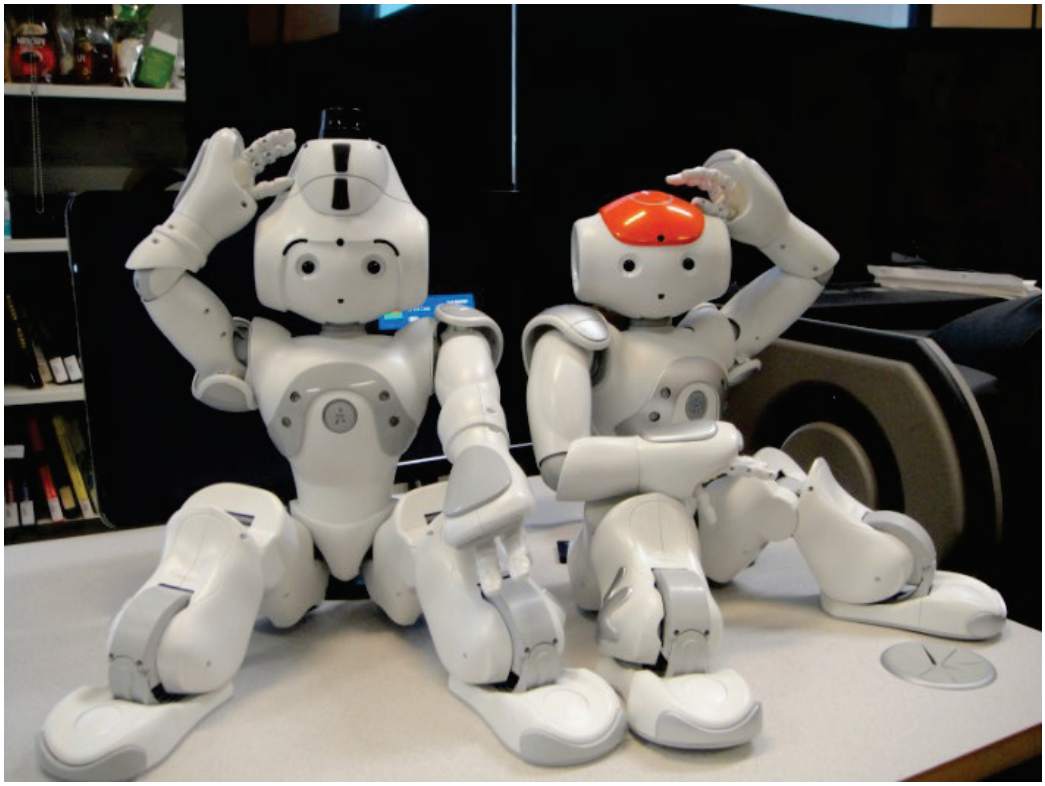


Figure 2.5. Humanoid Robot NAO in AISL lab

- Four-legged robots

A four-leg or tetrapod scheme is another walking system that is often found in nature. Comparing to a biped robot, a four-legged robot have the advantage of being more statically stable while standing. The walking pattern for a four-legged robot is designed in different ways including opposite pairs and alternating pairs [23].

- Six-Legged robots

Many of the walking robots have more than four legs for greater stability. Six legged (Hexapod) robot locomotion is the most popular leg locomotion because that it provides the static stability rather than a dynamic stability while moving and standing. Most of the six legged walking techniques are biologically inspired from insects such as six legs spiders. Wave gait and tripod gait are commonly used gait models for hexapods and robots with more legs [23].

- Waive gait [24]

As demonstrates in Figure2.6, waive gait has 5 steps:

1. All six legs in neutral position
2. Front pair of legs step forward
3. Second pair of legs step forward
4. Third pair of legs step forward
5. Robot body move forward following legs

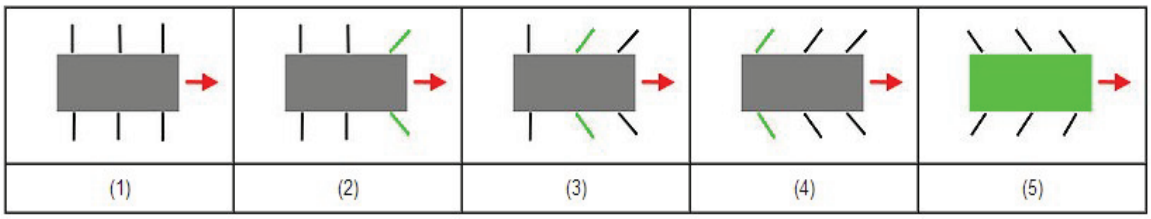


Figure 2.6. 5 steps wave gait [23]

- Tripod Gait [24]

Figure2.7indicates a 4 steps movement for tripod gait:

1. All six legs in neutral position
2. Alternating legs step forward on either side (3 legs)
3. The other 3 legs step forward
4. Robot body move forward following legs

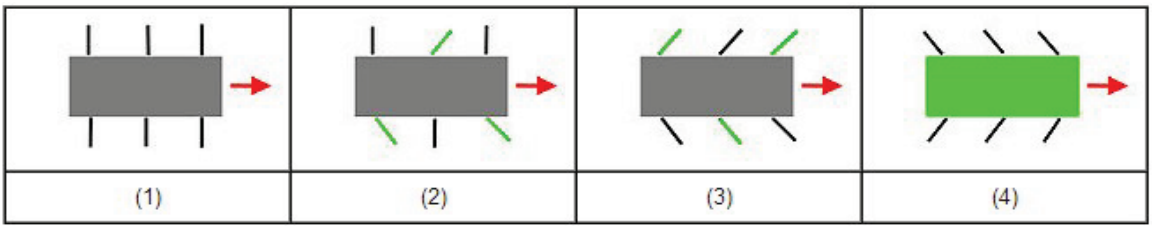


Figure 2.7. Tripod gait [23]

2.1.3. Comparison

Wheeled and legged locomotion have their advantages and disadvantages respectively. The preference of choice usually based on the purpose of use for the robot.

Table 2.1. Wheeled robot and Legged robot comparison

	Advantages	**Disadvantages**
Wheeled Robot	• Simple design, easy to program and maneuver [14] • Generally low-cost • Variety and customization for specific needs	• May lose traction • Limited contact area in common designs
Legged Robot	• Complicated design • High cost • Heavy and weak especially for many legs	• Complicated design • High cost • Heavy and weak especially for many legs

2.2. Mobile Robot Navigation Problem

The ability of achieving navigation is the fundamental requirement of autonomous mobile robots. Montello defines navigation as a “coordinated and goal-directed movement of one’s self (one’s body) through the environment” [25]. In other words, the task of mobile robot navigation is to guide the robot through the environment based on sensory information [26]. Mobile robot in a navigation task must be able to ask and answer three questions:

- Where am I?
- What does the world look like?

- How should I get there from my current location?

The first question is generally known as the robot self-localization, the second as map-building and map interpretation and the third question usually comes under the domain of path planning. The first two questions are principally concerned in this thesis and are considered as essential precursor to solving the remaining question of path planning.

2.2.1. Localization

In a robot navigation task, self-localization, also referred to as pose estimation, answers the question of where the robot is. The goal of localization is to keep track of the robot's current position and orientation with respect to a reference frame. The reference frame is usually defined by the initial position of the robot. A common solution to the mobile robot localization is providing with *a priori* map of the environment. The navigation task therefore becomes matching perceived environment features with elements of this map in order to estimate the location of the robot [27]. However, for navigation in an unknown environment, the *a priori* map is generally not provided. In this case, localization can be conducted in two methods: dead reckoning and external referencing [26]. In dead reckoning, the robot current position is measured using an internal sensor named odometry and the obtained robot pose is as relative to the previous pose. In external referencing, robot current position is determined based on sensing external landmarks. In addition, an effective approach of localization has been studied to fuse dead reckoning with external referencing based on metric reference model [28].

2.2.2. Map Building

Robotics map building enables a mobile robot to construct and maintain a model of its environment based on spatial information gathered incrementally [29]. Generally, the spatial information receives from the perceiving of the environment through external sensors and the internal sensors such as odometry provides the robot location information within the environment. Robot map building can be generally seen as a two-step process during navigation: first, the corresponding features from a new perception

of environment are identified and the robot's position is updated based on the found correspondences. Second, the corresponding features are updated to the spatial information of the environment to complete the map merging and updating. Methods for map building such as Geometric Approaches and Occupancy Grids are mentioned in [30].

2.2.3. SLAM

SLAM is an acronym for Simultaneous Localization and Mapping, which was originally invented by Durrant-Whyte and John J. Leonard [31] based on earlier work by Smith, Self and Cheeseman [32]. SLAM is a technique mostly used by mobile robots to build up a map within an unknown environment and while at the same time navigate through the environment using the map. SLAM is implemented onto NAO robot in my research project while the robot is placed in a provided environment with no *a priori* knowledge given.

In this section, we firstly address the problem of SLAM, explaining SLAM model, and then discuss the approaches used for solving SLAM. Attention is paid to probabilistic approach as this is the select method to complete SLAM task in my project.

SLAM Problem & Model

SLAM problem can be defined as follows: A mobile robot navigates through an unknown environment, beginning at a given location with known coordinates. As the robot roams around the environment, the uncertainty of the motion accumulates, making it increasingly more difficult to find its actual global coordinates. At the same time, the robot is able to sense its environments by recognizing certain particular features, i.e. Landmarks, through on-board sensors as it moves around. What makes SLAM a complex problem is that both the localization and mapping issues exist and they should be resolved simultaneously. Figure 2.8 illustrates the complete SLAM model in terms of its components and functionality.

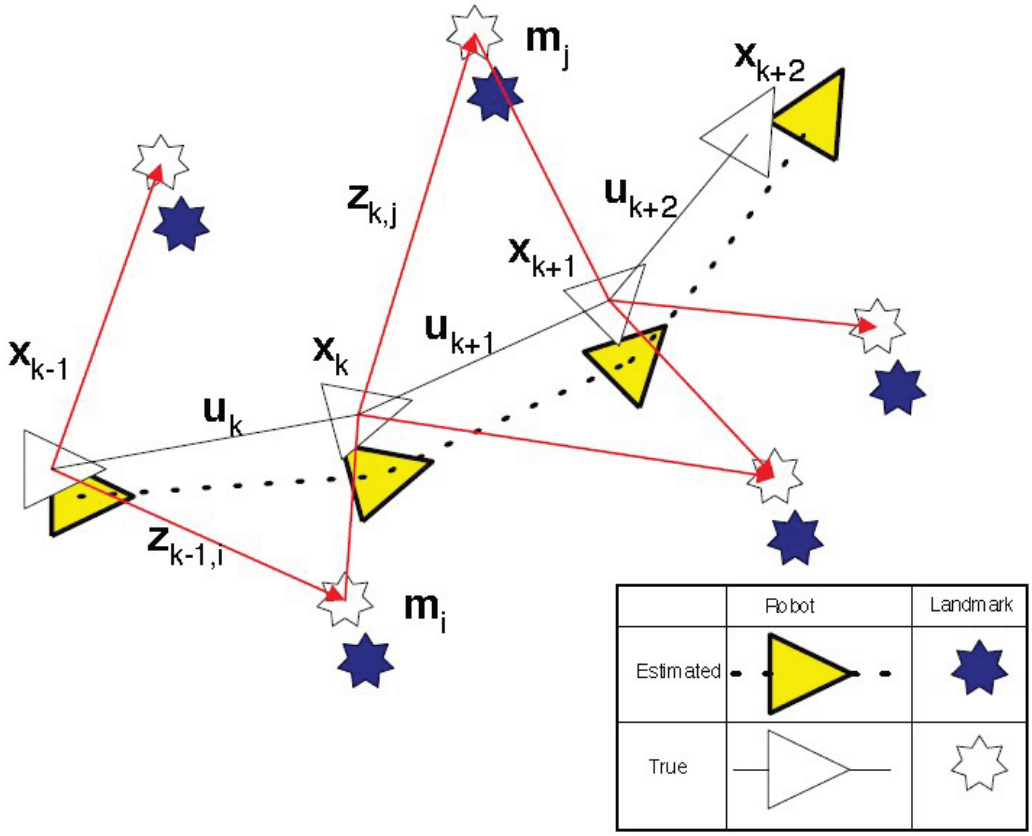


Figure 2.8. The essential SLAM problem [8]

Suppose a mobile robot navigates in an environment taking observations of a number of given landmarks using sensors mounted on the robot such as laser. The elements and terms of SLAM process, at an instant of time *k,* are described as follows. Note that the parameters used in the following explanations are shared with Figure 2.8.

- Robot position R

  This is also referred to as the system state vector. The sequence of robot locations is stored in this vector. For mobile robots on a 2D flat ground case, the matrix usually contains 3D matrix, including its 2D coordinate (x_r, y_r) in the space along with a sole rotational value θ_r for orientation. It can be given as:

$$R_k = \{R_0, R_1, ..., R_k\}$$

- Robot control motion U

  This refers to the control vector that is given to propel the robot in the prescribed directions. It is applied at time k-1 to drive the robot to the location x_k at time k. The control motion can be define as:

$$U_k = \{u_0, u_1, ..., u_k\}$$

- Map M

  In the case of landmark SLAM, the map vector stores all observed landmark 2D coordinate. Landmarks are captured by robot's external sensors as the robot moves around. For the case of environment with n landmarks, the Map vector M is described as:

$$M = \{m_1, m_2, ..., m_n\}$$

- Observation

  An observation is taken from the robot regarding the robot and landmark positions. The observation at time k is represented as z_k. Note that the robot may detect multiple landmarks at same time. It can be denoted by:

$$Z_k = \{z_1, z_2, ..., z_k\}$$

- Posterior

  A posterior refers to a set of vectors that contain the robot pose and all landmarks position, which can be written as:

$$X_k = [R_k, M_k] = [R_k, L_{1k}, L_{2k} \dots L_{nk}]$$

Solutions to the SLAM Problem

Since 1990s probabilistic approaches such as Kalman Filters, Particle Filter and Expectation Maximization have become dominant in solving SLAM problem and are discussed in the next sections. The main reason for using probabilistic technique is that robot localization and mapping is characterized by uncertainty and sensor noise. The probabilistic approaches manage the problem by modeling different sources of noise and their effects on the measurements [33].

In probabilistic SLAM, the uncertainty in motion and observation model from the robot is well represented in a probability distribution. A probability law rules the two main models in SLAM, the motion model and the observation model. The essential problem in SLAM is the calculation of posterior [34], and is commonly solved by two main methods in probabilistic SLAM, online SLAM and offline SLAM. At a time instant k, the online SLAM estimates the posterior probability of the current robot pose x_k with the map m based on observation and control motion data, Z_k & U_k respectively. It can be described by:

$$p(x_k, m | Z_k, U_k)$$

An offline SLAM, also referred to full SLAM, estimates the posterior probability of the overall previous path of the robot position, denoted as x_k, along with the map m, based the data from observation Z_k and control motion U_k similar to online SLAM:

$$p(x_k, m|Z_k, U_k)$$

Extended Kalman Filter based SLAM (EKF-SLAM)

Extended Kalman Filter based SLAM stems from Kalman Filter and is the most influential SLAM solutions among others. EKF–SLAM utilizes the extended Kalman filter (EKF), which is developed from Kalman filter (KF). The basic difference between KF and EKF is that the KF is only able to handle linear model, whereas the EKF is developed to handle nonlinear models and is more suitable to solve SLAM problem [8]. The EKF based SLAM method was introduced through a number of seminal papers [32, 35], and [31, 36, 37] includes the report regarding early implementation results.

In EKF-SLAM, the map M, usually called stochastic map, is a large vector containing sensors and landmarks states, and is modeled by a Gaussian variable [38]. As the robot moves, this map keeps updating by the EKF through two critical steps, prediction (robot motion model) and correction (the robot sensor detects the landmarks that had been mapped before). Additionally, in order to obtain a true exploration, the EKF-SLAM requires an additional step of landmark initialization, where the new landmarks are added into the map.

EKF-SLAM has a large range of applications in navigation problems such as airborne, underwater, indoor and other various types of robots [39]. Figure 2.9(a) demonstrates an underwater map, made by the state-of-the-art EKF-SLAM, obtained with the underwater robot named Oberon, from the University of Sydney, Australia, shown in Figure 2.9(b). The map in Figure 2.9(a) represents the robot trajectory, designated by the yellow triangles connected by a line. The ellipse near each triangle corresponds to the covariance of Kalman filter estimate relative to the robot pose. The size of ellipse is proportional to the uncertainty of robot current location. Red dots in this figure depict landmark detections, received by filtering the sonar scan for small and reflective objects. It is worth mentioning that the pattern of this EKF-SLAM plotting, in terms of how it represents each EKF-SLAM component and its characteristics, is

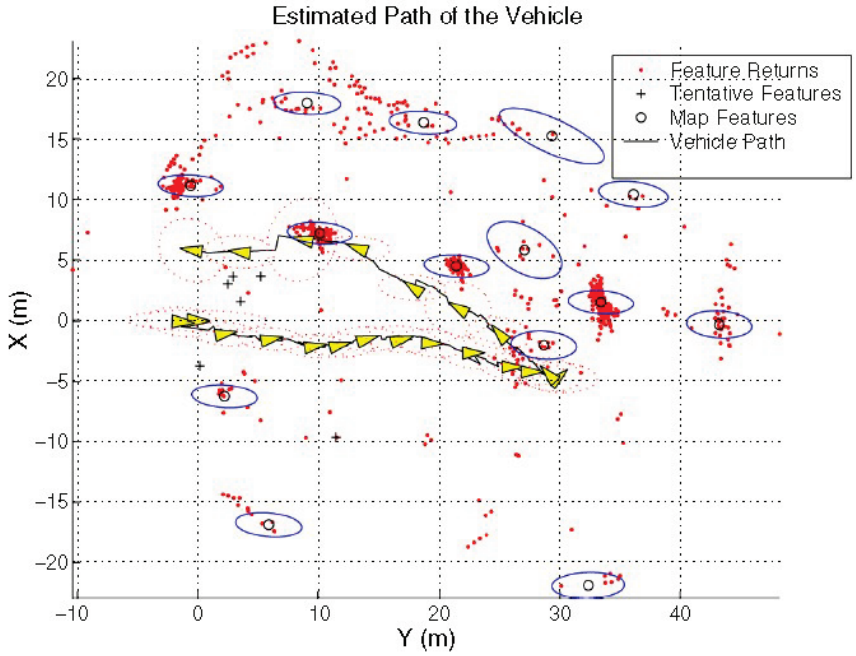
classically used for the demonstration of various types of robot explorations. Thus, most of the plotting results from my research project share the similar representation.

Particle Filter based SLAM (Fast-SLAM)

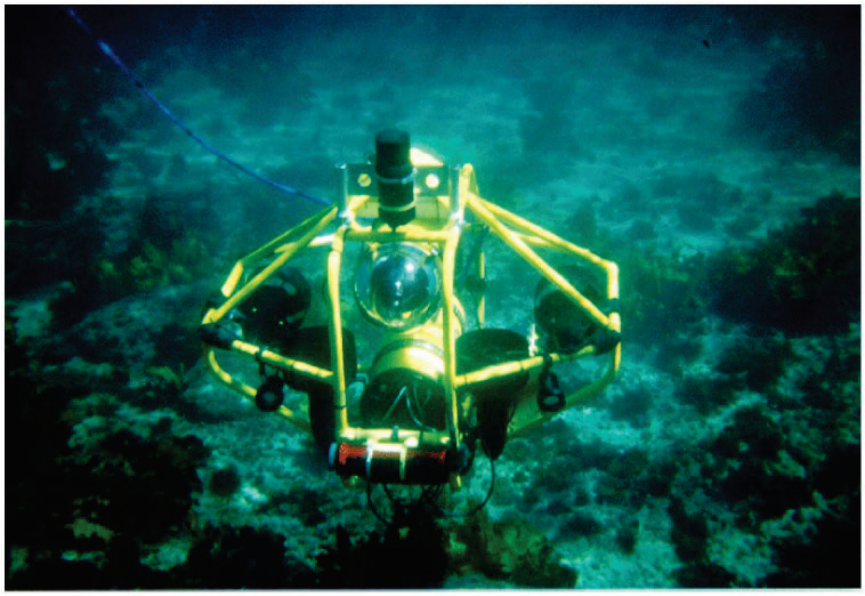
Particle Filter, also called the sequential Monte-Carlo (SMC) method, is a recursive Bayesian filter implemented in Monte Carlo simulations. This method executes SMC by random point clusters or also called “particles” to represent the Bayesian posterior [40]. Different from Extended Kalman Filter, Particle Filter (KF) draws a set of samples to represent the distribution. Such ability makes KF capable of handling highly nonlinear information and non-Gaussian noise. Nevertheless, this ability results in increase of computational demand on new landmark detection, what has limited it for real-time applications [41]. Fast-SLAM is one of the few works that combine PF with other techniques to solve SLAM problem. Such algorithm relies on the assumption of known data association and takes advantages of the idea that landmark estimations are conditionally independent given the robot’s path [42]. Each particle in Fast-SLAM makes its own local data association. In addition, less demand of computational expense than EKF and KF has been made on Fast-SLAM as it uses a particle filter to sample robot paths.

Expectation Maximization based SLAM

As an ideal option for map building rather than localization, Expectation Maximization is a statistical algorithm based on maximum likelihood (ML) estimation [40]. When the robot position is known, the EM algorithm is able to build the map by means of expectation [43]. The EM can be seen as a two-step iterated process: expectation step (E-step) and maximization step (M-step). In E-step, the posterior of robot positions is computed for a given map, while in M-step the most likely map is calculated according to given position expectations. As a result, the accuracy of map building increases. The advantage of EM over EKF is the great performance of dealing with data association problem [33]. In order to achieve that, the algorithm has to repeatedly localize the robot in the E-step to generate different possible correspondences. On the other hand, the fact that the repeat of processing same data for building the most likely map makes this algorithm inefficient and not suitable for real-time applications [44].



(a)



(b)

Figure 2.9. **(a) Sample of KF estimation of the map and robot position. (b) Underwater vehicle Oberon, developed at the University of Sydney [39]**

2.3. Augmented Reality

Augmented Reality has attracted research interests in many areas including medical, military, entertainment and etc. However, only a few applications are found in robotics research, especially in enhancing robotic navigation. This thesis proposes an approach of improving EKF-SLAM algorithm by integrating the technology of Augmented Reality. Prior to the mythology, a review of AR is useful to understand the knowledge.

2.3.1. Introduction of Augmented Reality

The fundamental idea of Augmented Reality (AR) is to mix the view of real environment with virtual or additional computer-generated graphic content in order to improve our perception of the surroundings. An example of AR application for mobile devices to obtain information in the environment is shown in Figure 2.10.

Figure 2.10. The user uses mobile devices to find "AR markers" in surrounding and obtain location information

Augment Reality is one part of more general area of mixed reality (MR) [45], which refers to a multi-axis spectrum of areas that cover Virtual Reality (VR), telepresence, Augmented Reality (AR) and other related technologies [46].

The term Virtual Reality is used for computer-generated 3D environments that allow the user to interact with synthetic environments [47-49]. VR users are able to enter a computers artificial world that can be a simulation of some form of reality or the simulation of a complex phenomenon [47, 50].

In telepresence, the goal is to extend user's problem solving abilities and sensory-motor facilities to a remote environment [46]. A good definition for telepresence is a human/machine system in which the human operator obtains sufficient information about the teleoperator and the task environment, displayed in a sufficiently natural way, that operator is provided the feeling of being in a remote location [51].

Augmented Reality can be seen as a technology between telepresence and Virtual Reality. The environment in telepresence is fully real and in VR is completely synthetic, in contrast, the user in AR is presented a real environment superimpose or "augmented" with virtual objects.

For a better understanding, AR systems can be defined by three classical and widely recognized criteria [52, 53]:

- Combines virtual and real

  AR requires display technology that allows the user to simultaneously see virtual and real information in a combined view. A see-through head-mounted display (HMD) is one of commonly used devices to combine real and virtual. The device lets the user see the real world, with virtual objects superimposed by optical or video technologies. Samples of HMD are shown in Figure 2.11.

Figure 2.11. Head-mounted Displays [11]

- Registered in 3-D

AR relies on an intimate coupling between the real and virtual that is based on their geometrical relationship. This makes it possible to render the virtual content with the right placement and 3D with respect to the real. For example, in the field of medicine, AR could guide precision tasks like where to perform a needle biopsy of a tiny tumor. Figure 2.12 shows a mock-up of a breast biopsy operation, where the 3D computer-generated graphics help to identify the location of the tumor and lead the needle to the target.

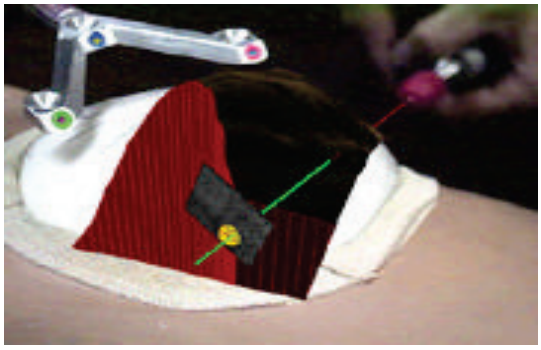


Figure 2.12. Mockup of breast tumor biopsy. 3-D graphics guide needle insertion [53]

- Interactive in real time

The AR system must run at interactive frame rates, such that it can superimpose the computer-generated information in real-time and allow user interaction. One example would be the implementation of AR on touch-screen human-computer interaction. Figure 2.13 shows a public space touch-screen interaction by sensing the position of knocking actions on glass surfaces from using acoustic tracker.

Figure 2.13. Touch-screen interaction in public spaces [52]

Further to above definitions, there are two other aspects that are necessary to mention. The definition does not limit to the sense of sight. AR is also able to apply to other senses of human being, including touch, hearing and smell. On the other hand, removing real objects by overlaying virtual ones, approaches known as mediated or diminished reality, is also considered AR [54].

2.3.2. Augmented Reality components

- Scene Generator

  The scene generator is the software or device that renders the scene. In current stage of AR technology, a few virtual objects need to be generated and they do not always have to be perfectly rendered to satisfy the purposes of the application [53], rendering is not one of the main problems.

- Tracking System

  The tracking system is one of the most difficult problems in AR systems because of the problem of registration [54]. In order to provide user a seamless combined view of virtual imagery and real objects, both worlds of real and virtual must be properly aligned with respect to each other. For example, many applications in the field of industry require precise registration, like medical systems [53, 55].

- Display

AR is still regarded as a developing technology and the solutions depend on different design purposes. Generally, AR display devices can be head-worn (retinal displays, miniature displays, and projectors), handheld (displays and projectors) or spatial (displays or projectors in the environment) [56].

There are two technologies available to combine the real and virtual world: optical and video technology. Each of them has certain advantages and disadvantages depending on factors like resolution, flexibility, field-of-view, registration and etc. [53].

Display technology is a factor that limits the development of AR systems. It is very difficult to find a see-through display that satisfies the requirements of resolution, brightness, field of view, and contrast [46] to present a seamlessly combined AR world. Furthermore, technologies that try to approach to these goals are still having the problem of size, weight and cost. Three classical AR display technologies are discussed in the next section.

2.3.3. AR Display Technologies

This section discusses three most classical AR display technologies, where optical see-through, video see-through and direct projection display systems are aim to overlay virtual objects to real world. The advantages and disadvantages are listed to provide a complete overview of these technologies. A table of summary is attached at the end of the section for comparison. For more about these technologies, please refer to [52, 57].

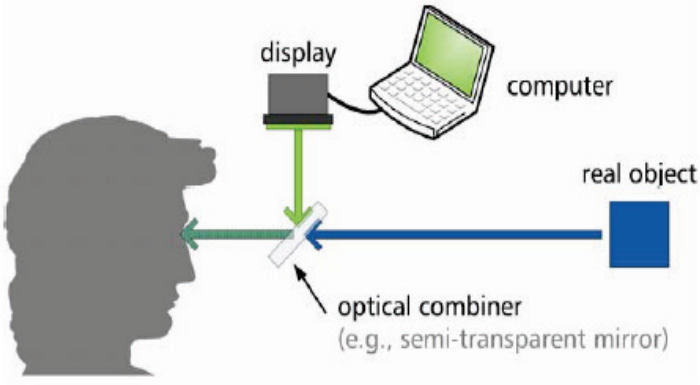


Figure 2.14. The optical path in an optical see through display system [52]

- Optical see-through displays

  Optical see-through devices work by using an optical combiner, for example, a holographic material or a half-slivered mirror [46]. The combiner is able to transmit the light from the real world environment and also reflecting the light from a computer display. Then, the optical combined light can be received by the user's eyes (Figure 2.14).

- Video see-through displays

  Video see-through displays technique is based on a camera that captures the view of the environment, a computer that generates virtual content, and a display that provides the combines view to user (Figure 2.15).

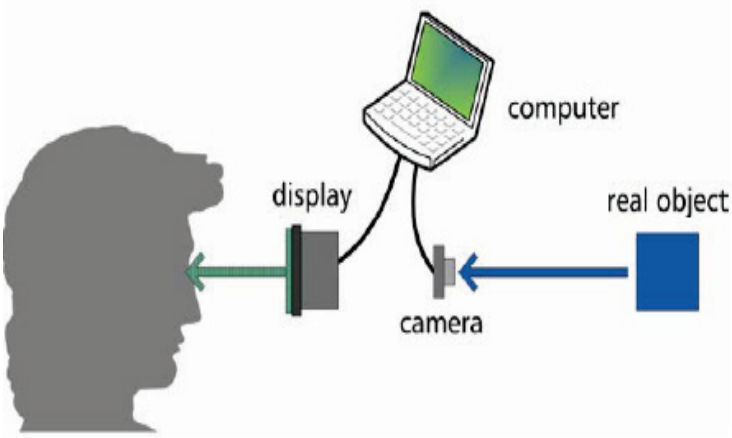


Figure 2.15. The optical path in a video see-through display system [52]

Head-worn displays can use video see-through techniques by placing cameras close to the eye positions. Ideally, two cameras should be used to acquire a stereo view, with one perspective for each eye, but monoscopic single-camera systems are common and easier to design and implement [58].

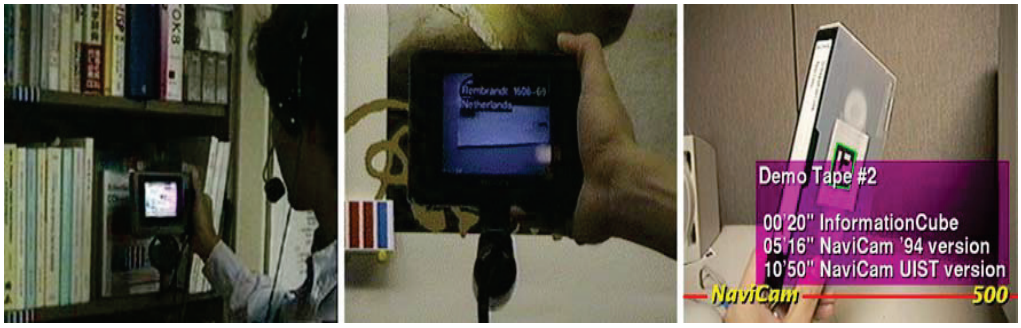


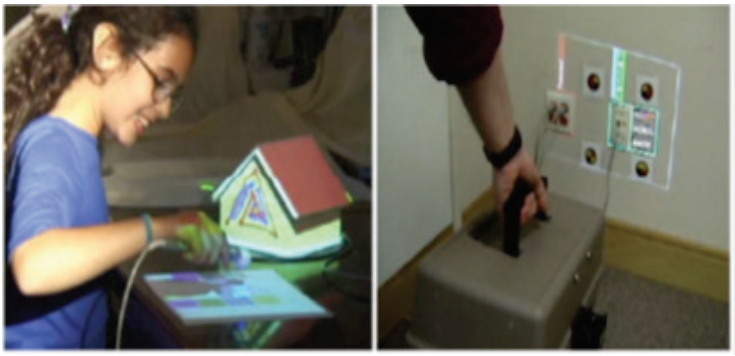
Figure 2.16. The video see-through display in NaviCam project [52]

Some video see-through displays use a camera to capture the scene, but present the combined view on a regular, typically handheld, computer display. A window-like effect, often referred to as a "magic lens," is achieved if the camera is attached on the back of the display, creating the illusion of see-through [59, 60]. Figure 2.16 illustrates how the camera on a handheld display can be used to recognize features in the environment, such that annotations can be overlaid onto the video feed.

Figure 2.17. Commercial camera phone achieving Video see-through AR [52]

Recently, the implementation of video see-through on mobile devices with built-in cameras has become more and more popular, In Figure 2.17, the camera located on the back of the device captures video of the real environment, which is

used by software on the device to recover the phone's pose related to tracked feature.



(a) (b)

Figure 2.18. Two examples of direct projection [52]

- Direct projection

Augmented Reality can also be achieved by directly projecting graphics onto the real environment. Figure 2.18 and 2.19 give examples of how the real world can be modified through controlled light that alters its appearance. Figure 2.18 (a) shows a child uses a tracked brush to apply virtual paint, which is projected onto physical objects and figure 2.18 (b) shows a handheld projector is combined with a camera that identifies elements of interest in the environment and augments them with projected light. In this example, a network socket is augmented with visualizations of network status and traffic. The Everywhere Displays project, as shown in figure 2.19, uses "steerable displays", where the system's projector can create augmentations on different surfaces in the environment, while the camera senses the user's interaction.

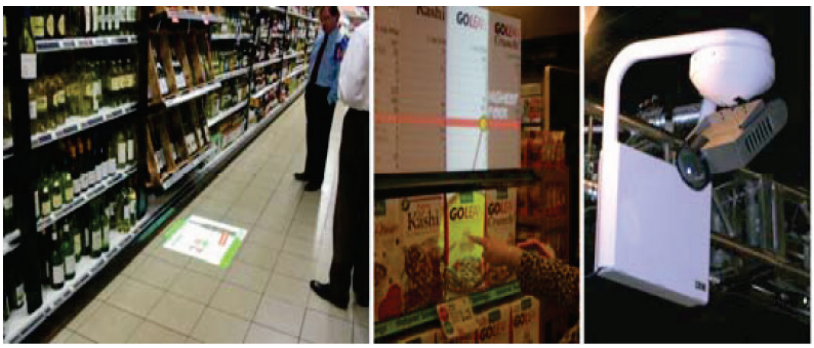


Figure 2.19. The Everywhere Displays project using "steerable displays [52]

2.3.4. Applications

Table 2.2. Summary of the advantages and disadvantages of display technologies

	Advantages	Disadvantages
3.1 Optical see-through	• Direct view of the real environment	• Lack of occlusion • Requiring advanced calibration and tracking • Reduced brightness
3.2 Video see-through	• Controlled combination of real and virtual	• Reduced quality and fidelity of the real environment • Potential perspective problems due to camera offset • Sensitivity to system delay • Dependency on camera operation
3.3 Direct projection	• Direct integration of the virtual with the real	• Dependence on environmental conditions • Dependence on projector properties

The technology of Augmented Reality has many possible applications in a wide range of areas. In this section, some of the fields are discussed, particularly emphasising AR for robotics, which is my research topic in the current stage.

Entertainment

AR not only can be applied in entertainment to build AR games, but also has improved the techniques of sports broadcasting and advertising.

- AR for Games

  Real world and computer games both have their own strengths. AR can be used to improve existing game styles and create new ones by combining the real and virtual contents to game world.

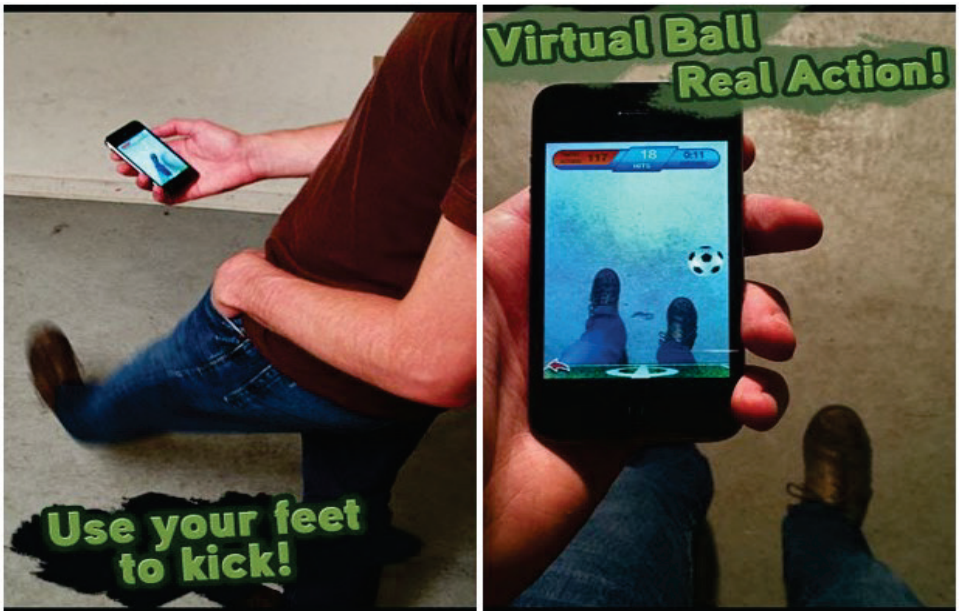


Figure 2.20. Player kicking the virtual football in AR Soccer

There are plenty of AR games running on smart phone platform, where iPhone has become one of the most popular one. The iPhone game “AR Soccer” creates a virtual football for player to kick through the camera Figure 2.20. Nintendo 3DS, a new generation of handheld game device, pre-installs one AR game named “Face Raiders”. The game will “capture” the player’s face and the goal is to shoot down all the enemies that have the player’s face on it (Figure 2.21).

Figure 2.21. 3DS game Face Raiders capturing faces

- AR for Sport Broadcasting

  Swimming pools, football fields, race tracks and other sports environments are well-known and easily prepared, in which video see-through augmentation through tracked camera feeds easy [57]. One example is the Fox-Trax system [61], used to highlight the location of a hard-to-see hockey puck as it moves rapidly across the ice, but AR is also applied to annotate racing cars (Figure 2.22a), snooker ball trajectories, life swimmer performances, etc. Thanks to predictable environments (uniformed players on a green and brown field) and chroma-keying techniques, the annotations are shown on the field and not on the players (Figure 2.22b).

(a) (b)

Figure 2.22. AR in life sports broadcasting: racing and football [54]

Maintenance

Complex machinery requires high level skill from maintenance personnel and AR has been found the potential in this area. For example, AR is able to automatically scan the surrounding environment with extra sensors to show the users the problem sites [57]. Friedrich [62] shows the intention to support electrical troubleshooting of vehicles at Ford and according to a Micro-Vision employee, Honda and Volvo ordered Nomad Expert Vision Technician systems to assist their technicians with vehicle history and repair information [63].

Medical Applications

Similar to maintenance personnel, doctors and nurses will obtain the benefits from critical information being delivered directly to their glasses [64]. Surgeons wearing AR devices can detect some features with naked eyes that they cannot see in MRI or CT scans [53]. An optical see-through augmentation is presented by Fuchs et al. [65] for laparoscopic surgery where the overlaid view of the laparoscopes inserted through small incisions is simulated (Figure 2.23).

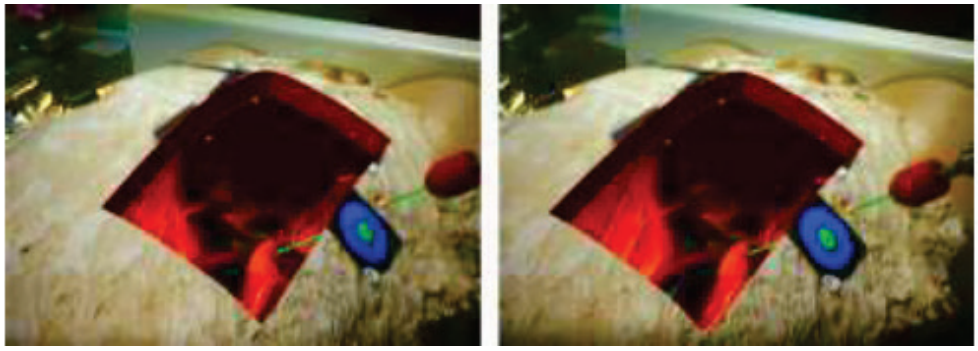


Figure 2.23. Simulated visualisation in laparoscopi [54]

Many AR techniques are developing for medicine use with live overlays of MR scans, CT, and ultrasound [57]. Navab et al. [66] already took advantage of the physical constraints of a C-arm x-ray machine to automatically calibrate the cameras with the machine and register the x-ray imagery with the real objects. Vogt et al. [61] uses video see-through HMD to overlay MR scans on heads and provide views of tool manipulation hidden beneath tissue and surfaces, while Merten [45] gives an impression of MR scans overlaid on feet (Figure 2.24).

Figure 2.24. AR overlay of a medical scan [54]

Military Training

For long time, the military has been using displays in cockpits that present information to the pilot on the windshield of the cockpit or the visor of the flight helmet [RW.ERROR - Unable to find reference:19] (Figure 2.25). For example, military aircraft

and helicopters have used Head-Up Displays (HUDs) and Helmet-Mounted Sights (HMS) to superimpose vector graphics upon the pilot's view of the real world. Besides providing basic navigation and flight information, these graphics are sometimes registered with targets in the environment, providing a way to aim the aircraft's weapons [53].

Figure 2.25. Military Training [54]

2.4. Augmented Reality in Robotics

Augmented Reality has been discovered for many years to improve the development of robot, such as robot navigation and Human and Robot Interaction (HRI). There are more and more studies have found on AR for humanoid robot. In following section, some of the applications of AR to humanoid robot are discussed.

2.4.1. Path Guiding

In the paper [67], AR is used for drawing guide paths to provide a simple and intuitive method for interactively directing the navigation of a humanoid robot through complex terrain. The AR user view of this method is as shown in Figure 2.26.

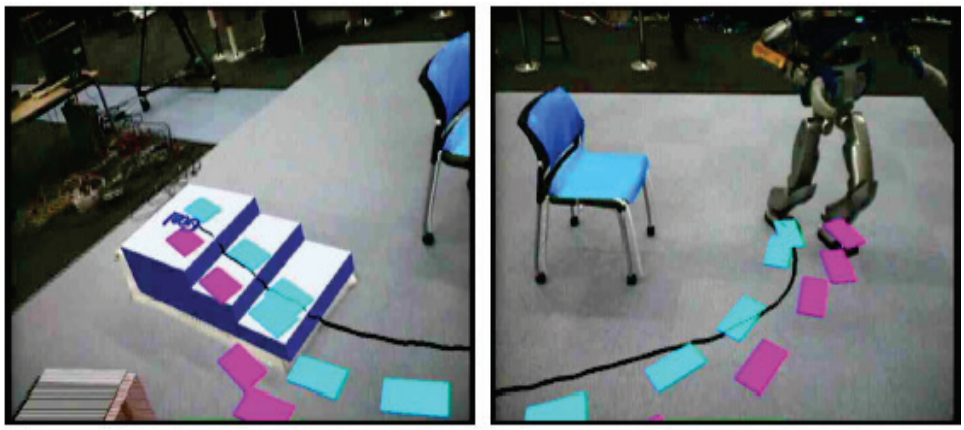


Figure 2.26. The augmented reality user view of the scene displaying the guide path and the robot's computed future footstep locations [67].

The user suggests an overall navigation route by drawing a path onto the environment while the robot is moving. The path is used by a footstep planner that searches for the suitable footstep locations which follow the assigned path as close as possible while respecting the robot dynamics and overall navigation safety. It has proven that the guidance provided by the human operator can assist the planner to find safe paths more quickly.

2.4.2. U-Tsu-Shi-O-Mi

U-Tsu-Shi-O-Mi shown in Figure 2.27 is an Augmented Reality system which consists of a synchronized pair of a humanoid robot and virtual avatars, and an HMD that overlay the avatars onto the robot. In this system, U-Tsu-Shi-O-Mi is an interactive AR humanoid robot that appears as a computer-generated character when viewed through special designed HMD. A virtual 3D avatar that moves in sync with the robot's actions is mapped onto the machine's green cloth skin (the skin functions as a green screen), and the sensor-equipped HMD tracks the angle and position of the viewer's head and constantly adjusted the angle that the avatar is displayed [68]. The result is an interactive virtual 3D character with a physical body that the viewer can literally reach out and touch.

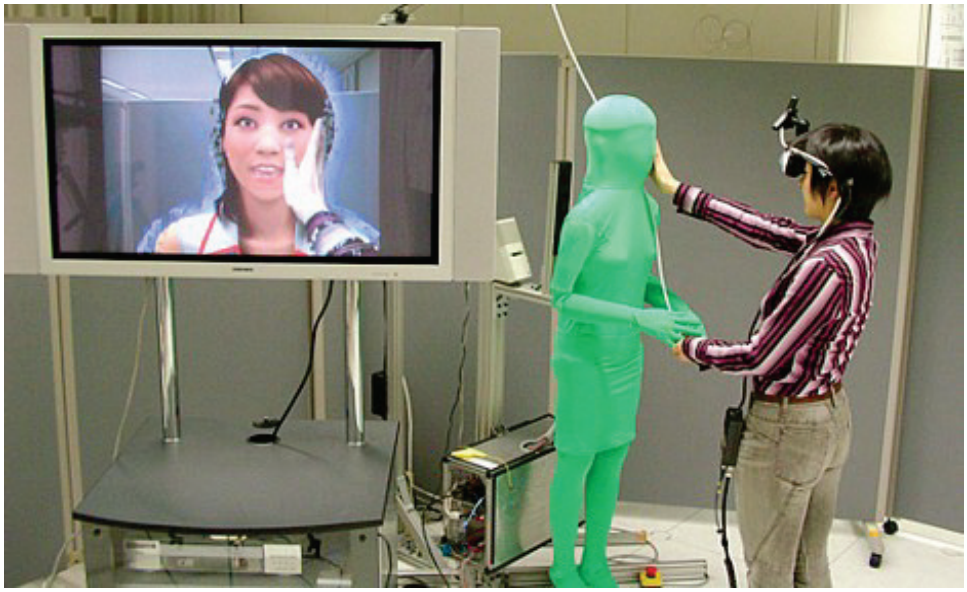


Figure 2.27. U-Tsu-Shi-O-Mi system [68]

2.4.3. Navigation in Unknown Environments

In this section, we introduce an application for vision-based localization system based on mobile augmented reality (MAR) and mobile audio augmented reality (MAAR) for both human and humanoid robot navigation in indoor environments.

Figure 2.28. AR markers to be placed in the environment

This application proceeds in two stages [10]: in the first stage, the designed system recognizes the location of a user from the image sequence taken from the environment using the system's camera and adds the location information to the user's view in the form of 3D objects and audio sounds with location information and navigation instruction content using Augmented Reality (AR). The information about the layout of environment and the location of AR markers are preloaded in the AR device such that the location can be recognized. Figure 2.28 gives the samples of AR markers.

A smart phone's camera and the marker detection make it possible to obtain the audio augmentation and 3D object placement can be achieved by the smart phone's operating processor and build-in graphical/audio modules. The task for the smart phone to detect AR marker is performed by using The ARToolKit, one of the pioneer software for making AR applications, which will be introduced in the next section.

Since the use of smart phone with camera in this navigation system replaces the components including mobile PC, wireless camera, head mounted displays (HMD) and a remote PC, the complexity of such system can be significantly reduced. This system is proving to have a wide range of applications and is capable of different purposes like museum tour guidance system and shopping assistance system.

Figure 2.29. Humanoid Robot NAO

In the next stage, the same AR module is transplanted onto a vision-based autonomous humanoid robot to determine the position with respect to its environment. The proposed technique is implemented on a humanoid robot NAO (Figure.2.29).

The navigation and localization performance is improved by presenting location-based information to the robot through different AR markers placed in the robot environment. Figure 2.30 demonstrates the outline of navigation strategy. The same AR navigation module will be used as a part of a visual simultaneous localization and mapping (Visual-SLAM) system, which is developing for the same humanoid robot platform [10].

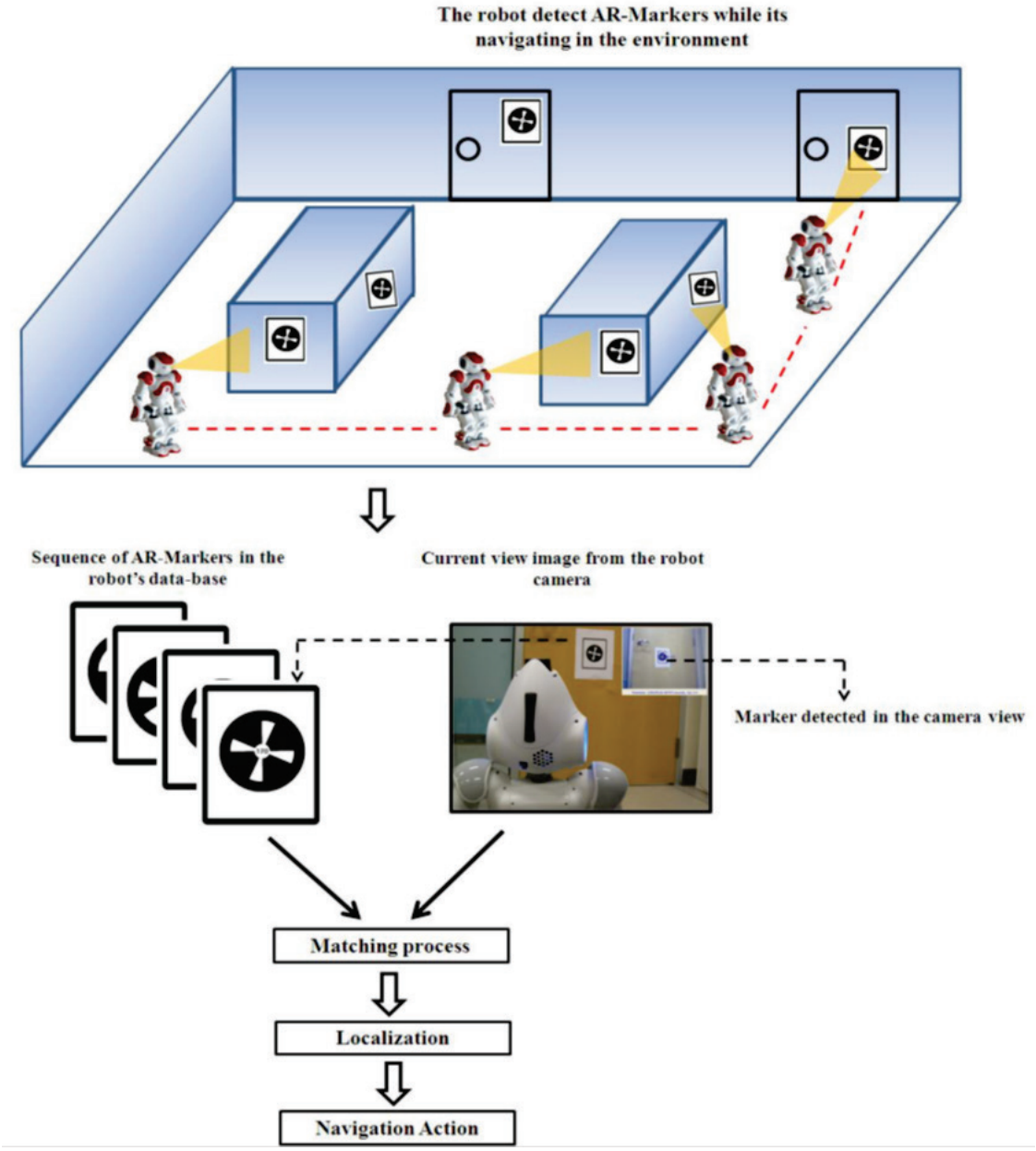


Figure 2.30. The outline of the navigation strategy using the data-base of AR-Markers

The fundamental idea of using camera to capture AR markers as landmark for SLAM navigation is adopted in my research project. As extension, an additional sensor, laser, is implemented to improve the performance and simplify the computational cost of SLAM algorithm, which is explained in the next section.

2.5. Summary

This chapter includes literature review on the topics of mobile robot, Augmented Reality and probabilistic approach in robot navigation. Discussing the three different topics in this chapter is to provide reader with comprehensive knowledge involved in my research project.

In the section of mobile robot, different types of mobile robot classified by the environment that the robot works are introduced. Among all types, attention is paid to ground based mobile robot as this is the most highly common type in the development of mobile robot. Wheeled and legged robots with multiple wheels/legs are then explained specifically in the subsection. It has been pointed out in this subsection that two-legged mobile robots have huge potential towards mimic human behaviors due to the similarity with human body structure. My research platform two-legged robot NAO is briefly introduced in this portion of review.

As the core technique implemented in my research project, an overview of Augmented Reality is included in the second section. Augmented Reality is a recent technology which enables user to obtain additional preloaded information from the observation of a particular object. This idea is implemented on my research project to improve the robot navigation based on EKF-SLAM algorithm. This section firstly demonstrates each component and display technologies that an Augmented Reality featured device commonly involves. Following portion of this section introduces AR applications in various fields. Three implementations of Augmented Reality in robotics are provided.

Simultaneous Localization and Mapping problem discussed in the last section provides a general presentation in terms of the formulation of and common solution of this problem. Main components in a SLAM problem are defined both in words and formulations. Two main methods in the probabilistic SLAM approaches, online SLAM and offline SLAM, are explained. This chapter has been concluded with a review of the most influential EKF-SLAM algorithm, which is the foundation of my research topic.

Implementation of Augmented Reality and EKF-SLAM has to be employed on a proper robot platform. In the next section, the chosen experimental platform humanoid robot NAO is introduced.

Chapter 3.

Robotic Platform NAO

The platform selected for my research project is humanoid robot NAO, a commonly used humanoid platform for education environment, produced by the French company Aldebaran Robotics [69]. NAO is a medium-sized humanoid robot developed mainly for universities and laboratories for research and education purposes. It replaced the Sony AIBO dogs in the RoboCup Standard Platform League (SPL) in 2008 [70]. As an autonomous humanoid robot, NAO is capable to move in a biped way, sense its close environment, communicate with human and think by on-board processor [71]. Figure 3.1 provides a summary of NAO robot and its main features including move, sense, communicate and think. In this chapter, an overview of NAO is presented, including NAO's hardware, mechanical and software architecture. Detailed interpretation based on my experimental implementation is provided. Programming in NAO is also included, which is then discussed in detailed in later chapter.

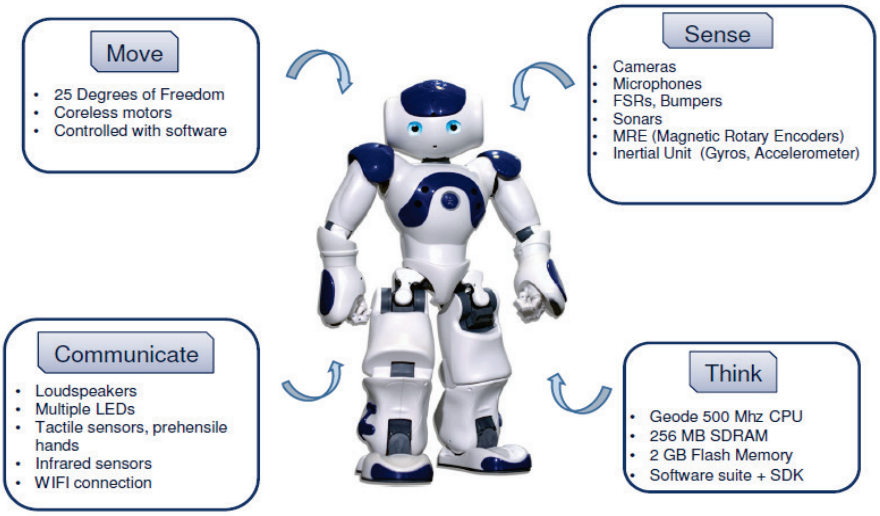


Figure 3.1. Humanoid Robot NAO and its main features

3.1. Hardware and Mechanical Architecture

3.1.1. Hardware

According to the Aldebaran's technical specification, NAO is 58 cm height and 4.3 kg weight, which is rather portable and light weight comparing to many other robot platforms. NAO is available in two versions: the standard version (Figure 3.1), and the laser head version (Figure 3.2). NAO with laser head is equipped in my research lab AISL as the laser sensor is essential for advance research such as SLAM. NAO's body is constructed from white technical plastic with some grey parts. NAO is powered by Lithium Polymer batteries offering between 45mn and 4 hours autonomy according to its activity. During my experiments, the battery was able to last for around 50mins when the robot performed regular load of activities involving a combination of sitting down standing up and walking work. Several sensor devices are equipped to obtain the information of its close environment.

Figure 3.2. NAO with laser head in AISL

3.1.2. NAO sensors

Sensors are essential for autonomous mobile robots to obtain information of surrounding environment in order to make decisions to complete the tasks. Table 3.1 provides the general classifications and types of sensors that are frequently used for

autonomous mobile robots. NAO robot has equipped with most of the popular sensors for either entry or advanced research requirements. NAO on-board sensors are bold marked in Table 3.1.

Table 3.1. General Classification of Robot sensors with NAO on-board sensor in bold

General classification	*Sensor*
Tactile sensors	• Contact switches, **bumpers** • Noncontact proximity sensors
Active ranging	• Reflectivity sensors • **Ultrasound sensor** • **Laser**
Localization in fix reference frame	• GPS • Active optical or RF beacons • Active ultrasonic beacons
Wheel/motor sensors	• Optical encoders • **Magnetic encoders** • Capacitive encoders • Inductive encoders
Heading sensors	• Compass • **Gyroscopes (IMU)**
Vision-based sensors (Cameras)	• CCD/**CMOS cameras** • Object tracking packages

- Ultrasound

NAO Robot has 2 sets of ultrasound devices(transmitter & receiver) situated in chest (Figure 3.3) that provide space information in 1 meter range distance if an object is situated at 30 degrees from the robot chest (60 degrees all cone combining both devices). The sonar sensor was utilized for the obstacle detection module on NAO. The sensor detects coming objects within the sonar range and then stops the robot.

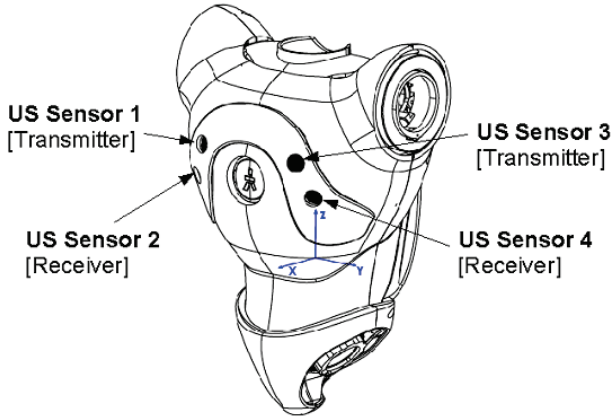


Figure 3.3. Ultrasound Sensors on NAO [70]

- Cameras

Two identical CMOS video cameras are located in the forehead as indicated in Figure 3.4. They provide a 640x480 resolution at 30 frames per second. They can be used to identify objects in the visual field such as goals and balls, and bottom camera can ease NAO's dribbles. The use of top camera is critical to my project as it is programmed to be able to detect specific NAO marks. It is found through the experiments that the camera running under 640x480 high resolution will result in great time-delay when the robot connected wirelessly with the computer. The resolution used in the experimented was therefore adjusted to 160x120.

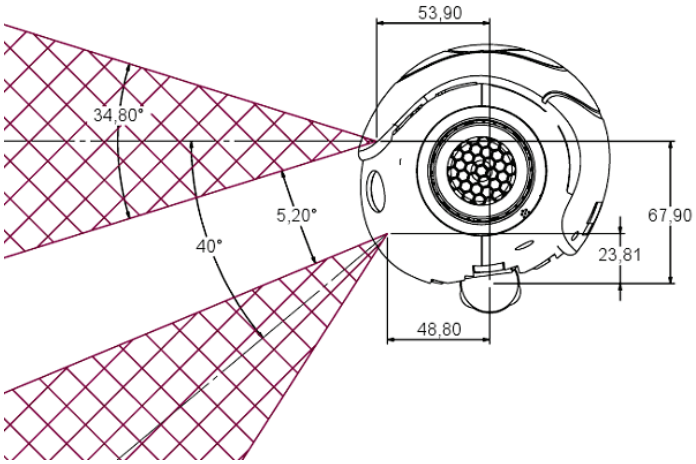


Figure 3.4. NAO Cameras [70]

- Microphone

  The NAO comes with 4 microphones at different parts on NAO's head. Microphones are very important sensors because we consider that voice should be the most natural interface between NAO and its users. NAO is capable of recognizing predefined voice command to carry out different tasks. It is noticed that the background noise severely impacts the quality of voice recognition. The experiment is recommended to conduct in a quite area for best accuracy.

- Bumper

  Bumper is a contact sensor that helps us know if robot is touching something, in this case the bumpers are situated in front of each NAO's foot and they can be used, for example, to know if the robot is kicking the ball or if there are some obstacles touching the feet. In my experiment, the bumper was used as a trigger to initialize the experiment.

- Force Sensors

  NAO has 8 Force Sensing Resistors (FSR) situated at sole of feet with 4 FSRs in each foot (Figure3.5). The value returned from each FSR is a time needed by a capacitor to charge depending on the FSR resistor value. It is not linear (1/X) and need to be calibrated. The sensors are useful when we are generating movement sequences to know if one position is a zero moment point (ZMP) and can be complements with inertial sensors. During the experiments, this sensor was found to be useful when NAO was taken off ground as robot walking. The FSRs detects the drop of force on the feet and pauses the action for hardware protection.

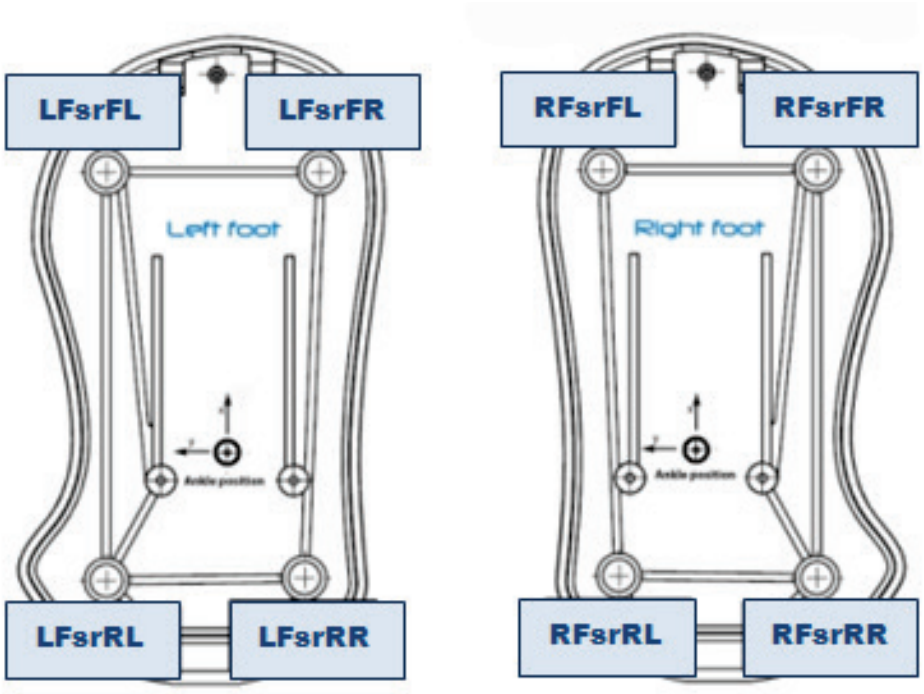


Figure 3.5. NAO FSR Sensors [70]

- Inertial measurement unit (IMU)

  NAO has 2 gyrometers in 1 axis and 1 accelerometer in 3 axes. These sensors are critical devices when we are working on precise motion Kinematics and Dynamics. They also help us to know if the robot is in a stable position while walking. Odometry of the robot can also be obtained from these sensors. In the preliminary stage of experiment, the gyrometers were intended to use for enhancing the build-in odometry of the robot. However, the gyrometer returns values in x, y axes and the z axis value, which is critical for representing robot orientation, is not accessible.

- Laser

  NAO in our lab is equipped with optional device laser head in order to feed our purpose in advanced research. This device is mounted on the center top of NAO's head as shown in Figure 3.6. Some specifications include detection range of *0.2m to 5.6m, angular* range *240°* started from *-120° to +120°*, laser wavelength 785nm, *0.36°* resolution, and refresh rate 100ms. 683 points can be detected within the coverage range by one scan from the laser sensor [70]. Besides the camera, the

laser sensor is another essential device for my research. The SLAM algorithm requires the laser to obtain landmark location information in terms of bearing and distance to perform a full SLAM process.

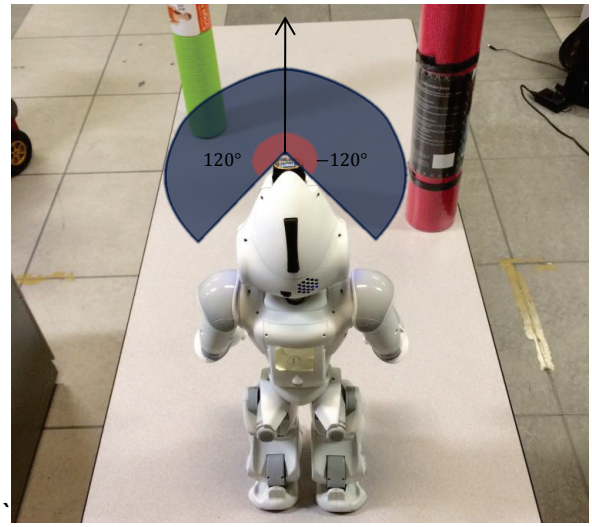


Figure 3.6. NAO Laser Head

3.1.3. Mechanical Architecture

Robot NAO has a total of 25 degrees of freedom (DOF), 11 degrees of freedom for the lower part of body including legs and pelvis, and other degrees of freedom for the upper part that includes trunk, arms and head. Following table gives the assignment of DOF for NAO [72].

Table 3.2. DOF on NAO

Total degrees of freedom (DOF): 25	
Head	2 DOF
Arms	5 DOF X 2
Pelvis	1 DOF
Leg	5 DOF X 2
Hands	1 DOF X 2

According to the Aldebaran NAO technical specification, each leg of NAO has 2 DOF at the ankle, 1 DOF at the knee and 2 DOF at the hip. The rotation axis of these two joints is 450 towards the body. Only one motor is needed to drive the pelvis

mechanism of NAO, which allows saving one motor at hip level without reducing the total mobility. In addition, each arm has two DOF at the shoulder, two DOF at the elbow, one DOF at the wrist and one DOF for the hand's grasping. The head is able to rotate about yaw and pitch axes. With the 25 DOF, the robot NAO is capable of performing various human-like behaviors.

3.2. Software Architecture

After introducing NAO hardware, the architecture of NAO software is discussed for a complete understanding of its software characteristics. Figure 3.7 provides a summary of NAO's software and their relations. The shade blocks indicate software used in this project. Figure 3.7 shows that Monitor, NAO SDK and Choregraphe as the software brought by Aldebaran, communicate with the NAOqi framework to obtain the access of various functions on NAO robot. NAO SDKs are programming packages in several computer languages used to meet the requirement of advanced research; this project was built based on NAO SDK for Python programming language. In next subsections, we present descriptions on NAO's software on each of them.

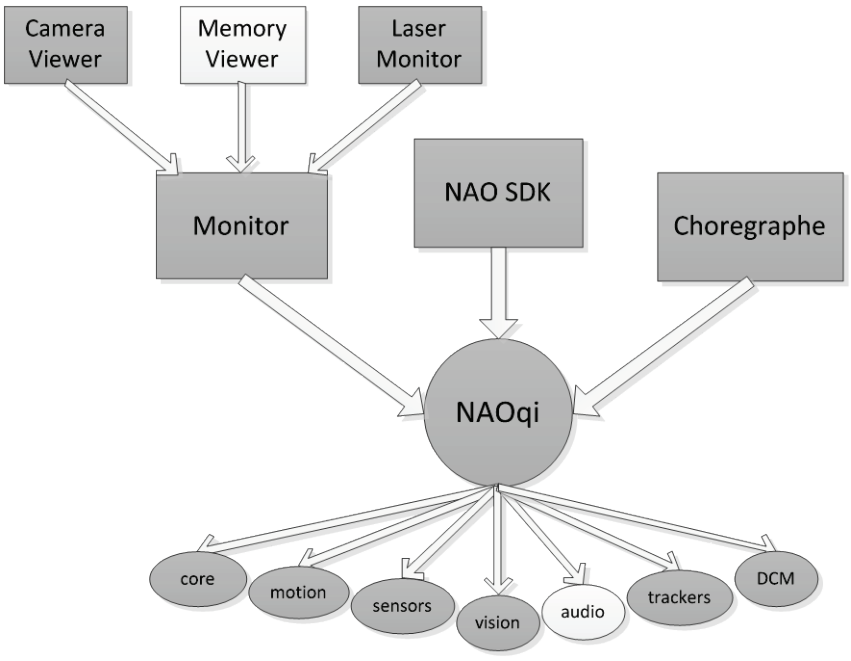


Figure 3.7. NAO Software Architecture

3.2.1. NAOqi

The main NAO software architecture is referred to as NAOqi by Aldebaran Robotics. NAOqi is designed as a distributed system where each component can be executed locally in robot's on-board system or be called remotely from another distributed system while NAOqi Daemon is running is the main system [73]. NAOqi is composed of three components: NAOqi OS, NAOqi Library and Device control Manager (DCM) shown in Figure 3.8

Figure 3.8. NAOqi components

The NAO OS also called OpenNao is an Open Embedded Linux Distribution modified to fit with NAO onboard system. Once the OpenNao is running in NAO's on-board system and the operation system initialization process is completed, NaoQi Daemon is triggered. NAOqi Library is divided in Python objects, also referred to modules. Each module has included some specific behavior that robot provides, i.e. walking, speaking. Modules that are required can be summoned through the main-broker [73]**.** The DCM, as in Device Control Manager, is similar to NaoQi library that is composed of several libraries for controlling the robot. As a difference, DCM controls the robot directly by sending calls to NAO's ARM controller, where ARM is the hardware architecture of NAO that monitors most of on-board motors. In addition, DCM is the essential part for the user to obtain the access to real-time image, or create a behavior of walking and reach a position.

The operating system that runs on NAO robot is embedded with Linux. Programming languages available for communication with NaoQi are C, C++, Python, Urbi and .Net. There are three NAO dedicated programs brought by Aldebaran

Company which are very useful for NAO developers. They will be discussed in the next subsection.

3.2.2. Choregraphe

Choregraphe is a user interface designed in an intuitive graphical environment that allows simple programming of NAO. It uses Python as the internal programming language. By dragging and dropping and connecting behaviors that are “packed” in boxes in the flow diagram style interface, NAO motions and sensor actions like walking, wave hands, text to speech, retrieve laser data, are easily performed. Choregraphe was implemented in the experiment for generating the pose for sit-down to walk initial, and the proposed SLAM algorithm took over. Figure3.98 shows the Choregraphe interface.

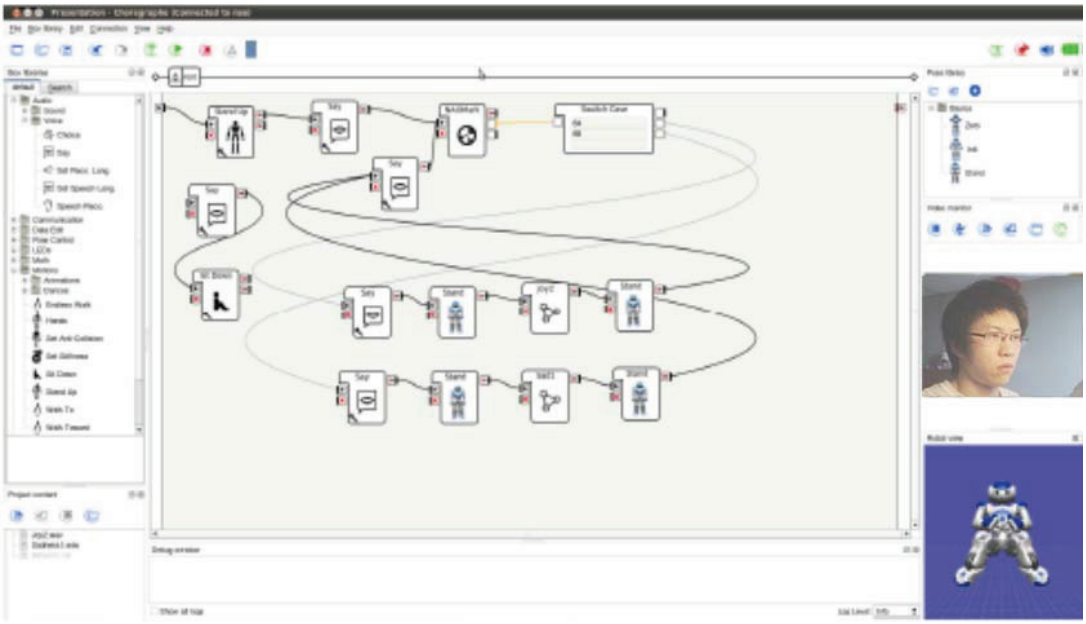


Figure 3.9. Building up NAO project by connecting “behaviour boxes”

3.2.3. Monitor

Monitor program in Figure 3.10 is composed of camera viewer, used for camera (stream a video, taking a picture, work with some embedded computer vision algorithms), Memory Viewer (view memory variables) and Laser Monitor (only work with NAO laser head). This software was useful for monitoring data in my experiment. For instance, the initial robot position or laser scanning range can be adjusted according to

the indication on laser monitor. On camera viewer, the program provides graphic indication superimposed on the live video based on the vision recognition function used.

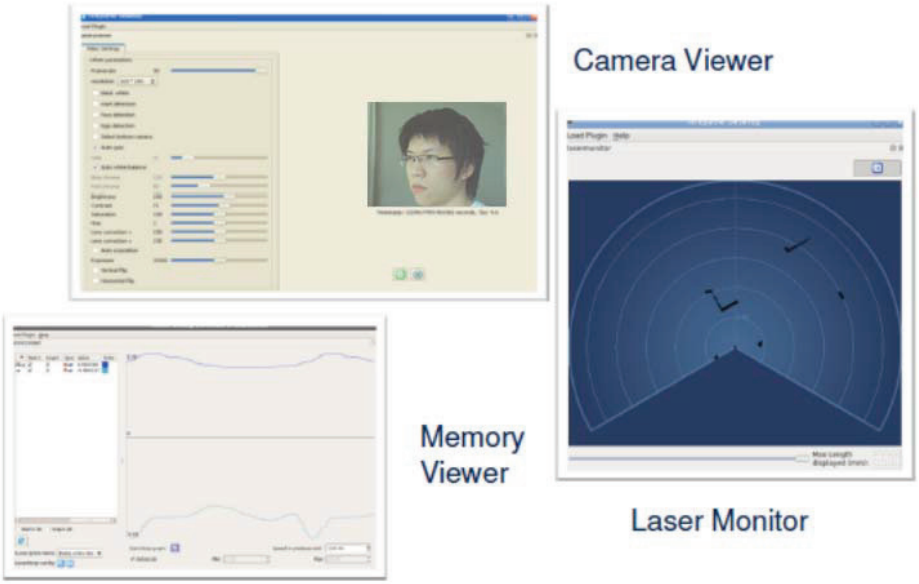


Figure 3.10. Monitor components

3.2.4. NAO Simulators

Simulation in robotics is critical as developers should test their program in a safe, virtual environment before any real time experiments. Simulation of NAO robot can be carried out through many simulators including one of the most well-known robot simulation software called Webots, brought by Cyberbotics Company. It recently released Webots for NAO, a dedicated version for the simulation of NAO robot, and the running window of the program is shown in Figure 3.11. Testing Most of the major behaviors from either Chorgraphe or Python script are supported. Although this simulator is very useful and can meet most of requirement for testing a NAO program, simulating laser function is yet to be supported.

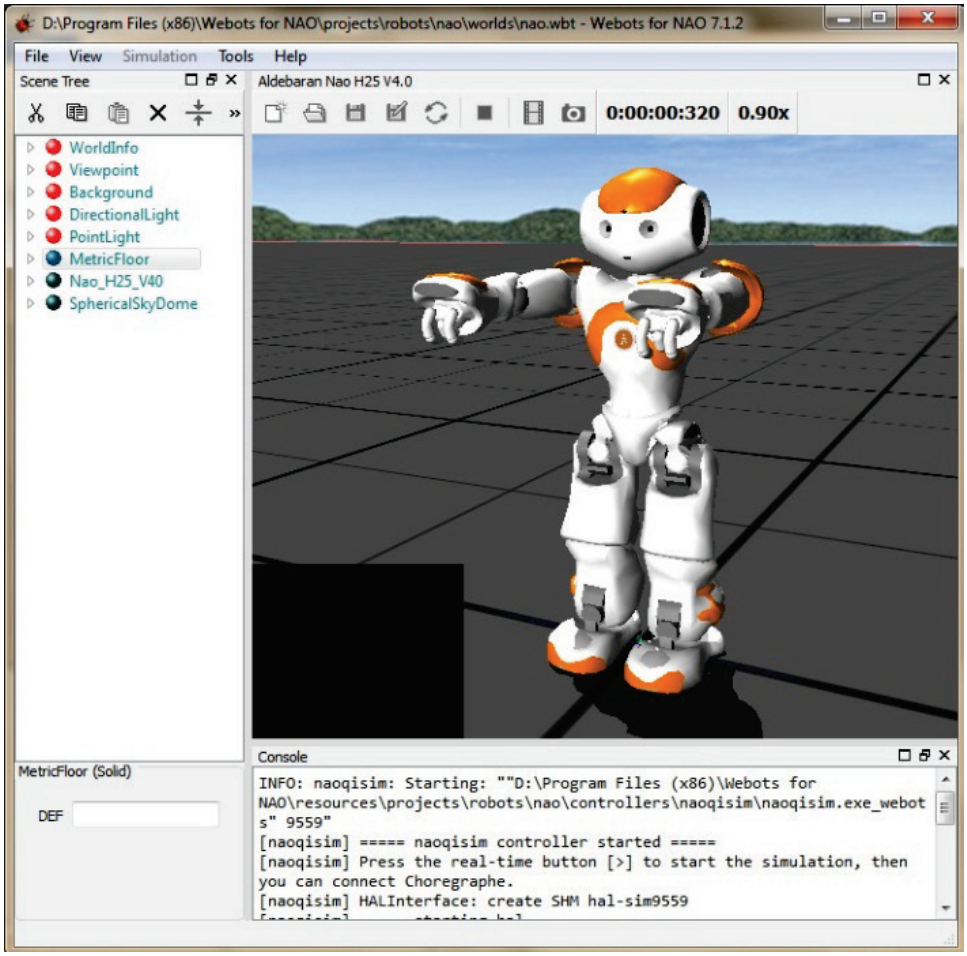


Figure 3.11. Webots for NAO Simulator

3.2.5. NAO Programming

Aldebaran provides several methods for develops to access NAOqi. Choregraphe permits an easy interface to use predefined NAO behaviors or design new behavior. As another approach for advanced developers, it is also possible to write the project in multiple supporting programming languages that include Python, C++, .NET etc. In my project, Python language is chosen to program NAO because it is highly compatible to NAO and Choregraphe, real-time supported and is also simpler to read and write than other programming methods. The characteristic for each selected methods is shown in the table below.

Table 3.3. Platforms to command NAO

Platform or languages	Running on	Tools	Remarks
Choregraphe	NAO Local	Choregraphe	Python code running locally on the robot
Python	NAO local & Remote control through Computer	Eclipse-Pythondev, Scite	Communications with the robot may be slow, Real-time is possible
C++	NAO local & Remote control through Computer	Visual Studio,Xcode.GCC Eclipse (Linux)	Cross compilation available on Linux (or Linux virtual machine), Real-time is possible
.NET	Remote control through Computer	Visual Studio	

3.2.6. Implementation method in the thesis

In this thesis, the implementation was achieved via the use of Choregraphe software and code in Python programming language. Choregraphe enabled NAO to be the suitable pose for the experiment, standing up, adjusting head level, etc. The SLAM algorithm written in Python and specifically modified for NAO robot was then executed to begin the experiment. Using Choregraphe in the experiment provides the secondary access to the robot in case of an experiment failure.

3.3. Summary

This chapter provides an overview about the NAO humanoid robot aiming to help understand this platform and my project. NAO robot is demonstrated generally by its hardware, mechanical architecture and software. The hardware section mainly includes several sensors as they are critical for the robot to explore the surrounding environment. The DOF distribution on NAO robot is listed and discussed in the mechanical architecture section. In the software, the structure of NAOqi is explained by the three components: NAOqi OS, NAOqi Library and Device control Manager. In addition, the dedicated NAO software, Choregraphe, Monitor and Webots, are demonstrated. The supported NAO programming languages for developing a NAO project are discussed.

Chapter 4.

EKF-SLAM Implementation

This chapter mainly discusses EKF-SLAM including a comprehensive description of the algorithm, as well as its simulation and real-time implementation. EKF-SLAM is interpreted with the description of SLAM components and SLAM process, and in the implementation section, simulated and real experiment results under the condition of different number of landmarks are demonstrated.

4.1. EKF-SLAM Algorithm

This section aims to provide readers a comprehensive description of landmark based EKF-SLAM algorithm that has been realized in my research project. We firstly introduce the frame transformation prior to the explanation of motion and observation model in a SLAM problem. Then the EKF-SLAM process is divided into three steps and discussed one by one.

4.1.1. Motion and Observation Models

In this subsection we present an explanation of the motion and observation models that are critical to solving the problem of SLAM and are also applicable to most of robot navigation problems. In addition, a brief recap of frame transformation is also included as this knowledge is frequently used in the algorithm.

Frame Transformation

There are two important frames that involve in a robot navigation problem, global frame, also referred to world frame, and robot frame, or named as local frame. The world frame is a fixed frame that keeps its origin at $\mathrm{G} = [0, 0, 0^{\circ}]'$, while the robot frame is

attached to the moveable robot at origin $\mathrm{R} = [0, 0, 0^{\circ}]'$. Global and robot frame can be transformed to each other by applying the translation $\mathrm{R}_{\mathrm{trans}} = [\mathrm{x}_{\mathrm{r}}, \mathrm{y}_{\mathrm{r}}]'$, and rotation R. Note that we focus on the frame transformation in the 2D plane due to the nature of the experimental environment.

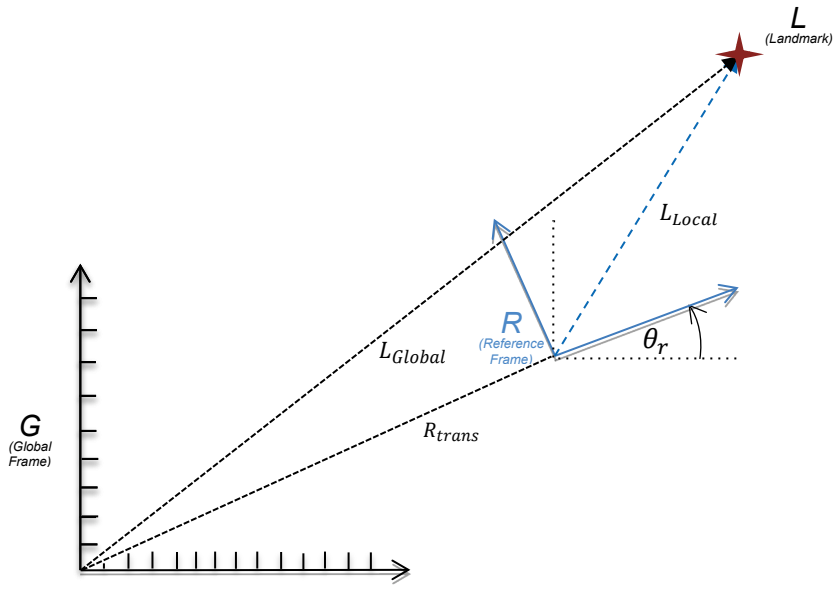


Figure 4.1. Transformation between global and local frame. Landmark is marked as red star.

Figure 4.1 above demonstrates the relationship of global and robot frame with a given landmark position. The Reference frame R is firstly rotate by θ_{r} and then translated by θ_r, from the global frame. What we need to derive is the transformation of the landmark position between global and local frame.

The transformation from global to local coordinate can be described as following equation:

$$\boldsymbol{L_{Local} = Rot^{T} * (L_{Global} - R_{trans})} \qquad \textbf{Equation 4.1}$$

Similarly, for the case of transformation from local to global frame:

$$\boldsymbol{L_{Global} = Rot * L_{Local} + R_{trans}} \quad \textbf{Equation 4.2}$$

Where $\boldsymbol{Rot} = \begin{pmatrix} cos\theta_r & -sin\theta_r \\ sin\theta_r & cos\theta_r \end{pmatrix}$ & $\boldsymbol{Rot}^T = \begin{pmatrix} cos\theta_r & sin\theta_r \\ -sin\theta_r & cos\theta_r \end{pmatrix}$ is the rotation matrix used when the robot frame only rotates around z-axis of the global frame by θ_r. Moreover, $L_{Local} = \begin{pmatrix} x_L \\ y_L \end{pmatrix}_{Local}$ is the landmark position with respect to local frame, and $R_{trans} = \begin{pmatrix} x_r \\ y_r \end{pmatrix}$ is the translation of local frame from global frame.

Motion Model

In the motion model, the current robot position at time t can be calculated according to a control motion u, a perturbation n ,and the last robot state $\boldsymbol{R}$ at time $t-1$. Thus, the motion model $f\ ()$ is denoted as:

$$\boldsymbol{R_t = f\ (R_{t-1}, u_t, n_t)} \quad \textbf{Equation 4.3}$$

The last robot state R_{t-1} is described by the translation and rotation with respect to the global frame as follow:

$$\boldsymbol{R_{t-1}} = \begin{pmatrix} \boldsymbol{R_{trans}} \\ \boldsymbol{\theta_r} \end{pmatrix}_{\boldsymbol{t-1}} = \begin{pmatrix} \boldsymbol{x_r} \\ \boldsymbol{y_r} \\ \boldsymbol{\theta_r} \end{pmatrix}_{\boldsymbol{t-1}} \quad \textbf{Equation 4.4}$$

The control motion u_t is represented as:

$$\boldsymbol{u_t} = \begin{pmatrix} \boldsymbol{u_{trans}} \\ \boldsymbol{\Delta\theta} \end{pmatrix}_{\boldsymbol{t}} = \begin{pmatrix} \boldsymbol{\Delta x} \\ \boldsymbol{\Delta y} \\ \boldsymbol{\Delta\theta} \end{pmatrix}_{\boldsymbol{t}} \quad \textbf{Equation 4.5}$$

The perturbation or noise n is also needed as the motion control in reality is never precise. Therefore, we added the noise characterized by Gaussian probability with q variance value and zero mean:

$$n = \mathcal{N}\ (0, q)$$

Direct & Inverse Observation Model

The observation model provides information regarding the relation between the robot and the landmark position. It is used when the robot observes a landmark that has

already been mapped by its on-board sensors. Direct observation $h()$ can be written as follow:

$$\boldsymbol{z = h(R, L_{Local}) = \begin{pmatrix} d \\ \emptyset \end{pmatrix} = \begin{pmatrix} \sqrt{(x_{L\_Loc} - x_r)^2 + (y_{L\_Loc} - y_r)^2} \\ \tan^{-1}\left(\frac{y_{L\_Loc} - y_r}{x_{L\_Loc} - x_r}\right) \end{pmatrix} + v} \qquad \textbf{Equation 4.6}$$

where d & $\emptyset$ are the distance and bearing between the robot and the landmark respectively. v represents a Gaussian noise vector with zero mean and s variance. Observation z is also referred to as measurement.

The inverse observation model $g()$ is called when there is a newly discovered landmark. Assume that the landmark measurement z is known, using inverse observation results the landmark state L with respect to the global frame. In most of cases, the function $g()$ is the inverse of observation function $h()$:

$$L = g(R, \mathrm{z})$$

4.1.2. EKF-SLAM Process

In general, the main essential process can be derived as three iterated steps [38, 74]:

1. Update the current state estimate using the odometry data (prediction step)
2. Update the estimated state from re-observing landmarks (correction step)
3. Add new landmarks to the current state (landmark initialization)

Prior to giving each step a detailed description in the following subsections, we will introduce the map state vector that is considered as the foundation of a SLAM algorithm.

The Map state

The map in a SLAM problem is a large estimated state vector storing robot and landmark states, which can be denoted as:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{M} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{l_1} \\ \boldsymbol{l_2} \\ \vdots \\ \boldsymbol{l_N} \end{bmatrix} \qquad \textbf{Equation 4.7}$$

Where R is the robot state containing position (x, y) and orientation θ, and M is the set of landmark positions $(l_1, l_2 \dots l_N)$, with N the number of current observed landmarks.

In EKF-SLAM, this map is modeled by a Gaussian variable obtaining from the mean and covariance $\boldsymbol{P}$ of the map state. The covariance matrix $\boldsymbol{P}$ is of importance to a SLAM problem as it describes the mean deviation and system uncertainty. The matrix contains the covariance on the robot position $\boldsymbol{P}_{RR}$, the landmarks, the covariance between robot position and landmark $\boldsymbol{P}_{RM}$ and its transpose $\boldsymbol{P}_{MR}$, as well as the covariance between the landmarks $\boldsymbol{P}_{MM}$, denoted as:

$$\boldsymbol{P} = \begin{bmatrix} [\boldsymbol{P}_{RR}]_{3\times3} & [\boldsymbol{P}_{RM}]_{3\times2N} \\ [\boldsymbol{P}_{MR}]_{2N\times3} & [\boldsymbol{P}_{MM}]_{2N\times2N} \end{bmatrix} = \begin{bmatrix} [P_{RR}]_{3\times3} & [P_{Rl_1}]_{3\times2} & \cdots & [P_{Rl_N}]_{3\times2} \\ [P_{l_1R}]_{2\times3} & [P_{l_1l_1}]_{2\times2} & \cdots & [P_{l_1l_N}]_{2\times2} \\ \vdots & \vdots & \ddots & \vdots \\ [P_{l_NR}]_{2\times3} & [P_{l_Nl_1}]_{2\times2} & \cdots & [P_{l_Nl_N}]_{2\times2} \end{bmatrix} \qquad \textbf{Equation 4.8}$$

Map Initialization

The mapping initializes with the initial position of the robot and no landmarks. Then we have the initial robot position as origin of the global frame, and the number of landmark n as zero. Consequently, the initial map state and covariance matrix can be considered as:

$$\boldsymbol{X} = \boldsymbol{R} = \begin{bmatrix} \boldsymbol{x_r} \\ \boldsymbol{y_r} \\ \boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \;\; \& \;\; \boldsymbol{P} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \qquad \textbf{Equation 4.9}$$

In SLAM, this map is continually being updated as the robot navigates through the environment. The dimension of the map state vector and covariance matrix will increase as new features recognized by the robot. i.e. Observing new landmark.

Robot Motion (Prediction step)

The EKF-SLAM algorithm in this first step calculates the position of the robot after the movement given by a control motion u. Due to the fact that only the robot position is changed in this step, the affected elements in the map state are those related to robot pose R, and the landmark part M remains invariant. Notice that the new robot pose here is derived through motion model function $f\ ()$ that has been discussed earlier. Therefore, the new map state can be written as:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{M} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}\ (\boldsymbol{R}, \boldsymbol{u}, \boldsymbol{n}) \\ \boldsymbol{M} \end{bmatrix} \qquad \textbf{Equation 4.10}$$

Where the new robot state R is based on the last robot position, control motion u and noise n, which can be expended as:

$$\boldsymbol{R} = \boldsymbol{f}\ (\boldsymbol{R}, \boldsymbol{u}, \boldsymbol{n}) = \begin{bmatrix} \boldsymbol{x_r} + \Delta \boldsymbol{x c\theta_r} - \Delta \boldsymbol{y s\theta_r} \\ \boldsymbol{y_r} + \Delta \boldsymbol{x s\theta_r} + \Delta \boldsymbol{y c\theta_r} \\ \boldsymbol{\theta_r} + \Delta \boldsymbol{\theta} \end{bmatrix} + \boldsymbol{n} \qquad \textbf{Equation 4.11}$$

The computation of covariance is completed, according to following equation:

$$\boldsymbol{P} = \boldsymbol{FPF^T} + \boldsymbol{F_n Q F_n^T} \qquad \textbf{Equation 4.12}$$

With $\boldsymbol{Q}$ the covariance matrix of the noise n, and where $\boldsymbol{F}$ and $\boldsymbol{F_n}$ are Jacobian matrices of motion function $f\ ()$.

$$\boldsymbol{F} = \begin{bmatrix} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{R}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \quad \& \quad \boldsymbol{F_n} = \begin{bmatrix} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \\ \boldsymbol{0} \end{bmatrix} \qquad \textbf{Equation 4.13}$$

Note that most of the parts in the above matrices are zero and identity as the large part of the map is invariant upon robot motion. As a consequence, minor parts are updated including$\boldsymbol{P}_{RM}$, $\boldsymbol{P}_{MR}$ and $\boldsymbol{P}_{RR}$ in the covariance square matrix $\boldsymbol{P}$, and the robot state $\boldsymbol{R}$ in the map state X. Updated parts are marked in gray.

$$X = \begin{bmatrix} R \\ l_1 \\ l_2 \\ \vdots \\ l_N \end{bmatrix} \quad \& \quad P = \begin{bmatrix} \left[P_{RR}\right]_{3\times3} & \left[P_{Rl_1}\right]_{3\times2} & \cdots & \left[P_{Rl_N}\right]_{3\times2} \\ \left[P_{l_1R}\right]_{2\times3} & \left[P_{l_1l_1}\right]_{2\times2} & \cdots & \left[P_{l_1l_N}\right]_{2\times2} \\ \vdots & \vdots & \ddots & \vdots \\ \left[P_{l_NR}\right]_{2\times3} & \left[P_{l_Nl_1}\right]_{2\times2} & \cdots & \left[P_{l_Nl_N}\right]_{2\times2} \end{bmatrix}$$

The psesudo code for pridiction step in the EKF-SLAM experiment is shown as follow:

- Robot move
- $\boldsymbol{R}$ = get measurement from robot odometry
- Compute covariance $\boldsymbol{P}$
- Update map state $\boldsymbol{X}$ and covarance matrix $\boldsymbol{P}$

Observation of mapped landmarks (Correction step)

The observation step occurs when a previously mapped landmark is measured by the embarked sensor on the robot. Once the data is collected from the sensor, the observation model $h()$ is used to calculate the innovation, which is basically the difference between the predicted and actual observation, used to reduce the uncertainty of the map state from the prediction step.

The innovation is also critical to the so-called data association problem in SLAM. The problem arises when the robot has to determine whether a detected landmark corresponds to a previously observed landmark or to a new one [75]. The description of the method used to solve the data association problem is included in this subsection.

Accordingly, the innovation vector z_{innov} is calculated from the following equation:

$$\boldsymbol{z_{innov} = z_{actual} - z_{predicted}} \qquad \textbf{Equation 4.14}$$

The innovation covariance Z_{innov} is then computed to measure the uncertainty of predicted observation,

$$\boldsymbol{Z_{innov} = H_X P H_X^T + S} \qquad \textbf{Equation 4.15}$$

Where $\boldsymbol{H_X}$ is the jacobian of the predicted observation model $z_{predicted}$. The structure of $\boldsymbol{H_X}$ is presented as:

$$\boldsymbol{H_X = \left[[H_R] \;\; 0 \;\; \ldots \;\; 0 \;\; \left[H_{L_{i\_Global}}\right] \;\; 0 \;\; \ldots \;\; 0 \right]_{2\times(3+2N)}} \qquad \textbf{Equation 4.16}$$

Where

$$\boldsymbol{H_R = \frac{\partial h}{\partial R}} \;\&\; \boldsymbol{H_{L_{i\_Global}} = \frac{\partial h}{\partial L_{i\_Global}}} \qquad \textbf{Equation 4.17}$$

With h the direct observation, $\boldsymbol{R}$ the robot state, and L_{i_Global} the predicted landmark state.

Since the computation of the innovation Jacobian matrix $\boldsymbol{H_X}$ is sparse, it only involves the robot state $\boldsymbol{R}$, the concerned landmark state $\boldsymbol{L_i}$ and their covariance $\boldsymbol{P}_{RR}$ $\boldsymbol{P}_{L_iL_i}$ along with their cross-variances $\boldsymbol{P}_{L_iR}$ and $\boldsymbol{P}_{RL_i}$.The representation of involved parts (marked as grey) in the covariance matrix $\boldsymbol{P}$ is:

$$\boldsymbol{X} = \begin{bmatrix} R \\ \vdots \\ L_i \\ \vdots \end{bmatrix} \;\&\; \boldsymbol{P} = \begin{bmatrix} [P_{RR}]_{3\times3} & \cdots & [P_{RL_i}]_{3\times2} & \cdots \\ \vdots & \ddots & \vdots & \ddots \\ [P_{L_iR}]_{2\times3} & \cdots & [P_{L_iL_i}]_{2\times2} & \cdots \\ \vdots & \ddots & \vdots & \ddots \end{bmatrix}$$

The map state X and the covariance P then need to be updated to complete the correction step. In order to do this, a Kalman gain K from the EKF algorithm has to be computed using the following formula:

$$\boldsymbol{K = PH_X^T Z_{innov}^{-1}} \qquad \textbf{Equation 4.18}$$

Notice that the Kalman gain matrix K contains a set of numbers about how much each of the robot state and landmark state should be updated.

Consequently, the full map state is updated because that Kalman gain affects the full state:

$$\boldsymbol{X = X + Kz_{innov}} \qquad \textbf{Equation 4.19}$$

Similar to the covariance matrix:

$$\boldsymbol{P = P - KZ_{innov}K^T} \qquad \textbf{Equation 4.20}$$

The psesudo-code for correction step in the EKF-SLAM process is shown as below:

- *Get measurement from laser sensor*
- *If this is observed landmark:*
  - *Compute expected lanmark position using* $h()$
  - *Compute innovation* $\boldsymbol{z_{innov}}$ *and its convariance* $\boldsymbol{Z_{innov}}$
  - *Computer Kalman Gain* K
  - *Update map state* $\boldsymbol{X}$ *and covarance matrix* $\boldsymbol{P}$

Data Association

Data association is one of the main challenges in the SLAM problem. In this thesis, it is handled by adopting Mahalanobis distance (MD) gating approach [76], where MD represents the probabilistic distance between the actual and estimated observations in EKF-SLAM:

$$\boldsymbol{MD^2 = z_{innov}^T Z_{innov}^{-1} z_{innov}} \qquad \textbf{Equation 4.21}$$

Where z_{innov} is the innovation vector and Z_{innov} the covariance matrix of innovation.

MD^2 is then used to compare with the gate validation scalar threshold σ^2. If the value of MD^2 is less than the validation gate σ^2, the detected landmark will be determined as the re-observed one and the correction step will begin. For the case that the MD^2 is greater than the σ^2, the landmark will be considered as new and initiate the landmark initialization step that is clarified in the next section.

The process of data association is descripted as pseudo-code:

- Compute $\boldsymbol{MD^2}$ according to innovation $\boldsymbol{z_{innov}}$ and covariance $\boldsymbol{Z_{innov}}$
- If $MD^2 < \boldsymbol{\sigma^2}$:
  - Observed landmark, go to correction step
- Else:
  - New landmark, go to landmark initialization step

Landmark Initialization Step

The step of landmark initialization only happens when the robot detects a landmark that is not yet observed and decides to add it in the map. As a result, the size of map state X and covariance matrix P is increased. This step is considered to be relatively straightforward as we only need to use the inverse observation function $g\ ()$ to compute the new landmark state L_{N+1} and add it into the map state X and covariance matrix P.

Assume that N indicates the number of mapped landmark and the observation for the new landmark at a time instant t is z_{N+1}, by using the inverse observation function $g\ ()$, we obtain the new landmark coordinated L_{N+1} with respect to the global frame G:

$$\boldsymbol{L_{N+1} = g(R_t, z_{new})} \qquad \textbf{Equation 4.22}$$

Next, this additional landmark L_{N+1} is added to the map state X:

$$X = \begin{bmatrix} R \\ M \\ L_{N+1} \end{bmatrix}$$

Also the covariance matrix P is augmented:

$$P = \begin{bmatrix} P & [P_{XL}]_{3\times2} \\ [P_{LX}]_{2\times3} & [P_{LL}]_{2\times2} \end{bmatrix}$$

Which includes landmark's co-variance P_{LL} and cross-variance P_{LX}:

$$P_{LL} = G_R P_{RR} G_R^T + G_z S G_z^T$$

$$P_{LX} = G_R P_{RX}$$

With S the covariance matrix of the observation noise v, and where G_R and G_z are Jacobian matrices of inverse observation $g()$:

$$G_R = \left.\frac{\partial g}{\partial R}\right|_{X_t, z_{N+1}} \quad \& \quad G_z = \left.\frac{\partial g}{\partial z}\right|_{X_t, z_{N+1}}$$

Following representation shows the appended parts to the map state X and covariance matrix P in the landmark initialization step. The appended parts (marked in grey) contain the new landmark's state in the full map state and its covariance and cross-variance in the covariance matrix P:

$$X = \begin{bmatrix} R \\ l_1 \\ l_2 \\ \vdots \\ l_{N+1} \end{bmatrix} \quad \& \quad P = \begin{bmatrix} [P_{RR}]_{3\times3} & [P_{Rl_1}]_{3\times2} & \cdots & [P_{Rl_{N+1}}]_{3\times2} \\ [P_{l_1R}]_{2\times3} & [P_{l_1l_1}]_{2\times2} & \cdots & [P_{l_1l_{N+1}}]_{2\times2} \\ \vdots & \vdots & \ddots & \vdots \\ [P_{l_{N+1}R}]_{2\times3} & [P_{l_{N+1}l_1}]_{2\times2} & \cdots & [P_{l_{N+1}l_{N+1}}]_{2\times2} \end{bmatrix}$$

Following list presents the pseudo-code for the landmark initialization step.

- Get measurement from laser sensor
- If this is new landmark
  - Compute lanmark position in global frame using $g()$
  - Compute covariance $\boldsymbol{P}$
  - Augmented map state $\boldsymbol{X}$ and covarance matrix $\boldsymbol{P}$ with new landmark

Once the landmark Initialization step is completed, the SLAM algorithm is ready for the next iteration. The robot will move again, observe landmarks, go through the step of correction or landmark initialization based on the decision of data association. The flow chart illustrating the full EKF-SLAM process is shown in Figure 4.2.

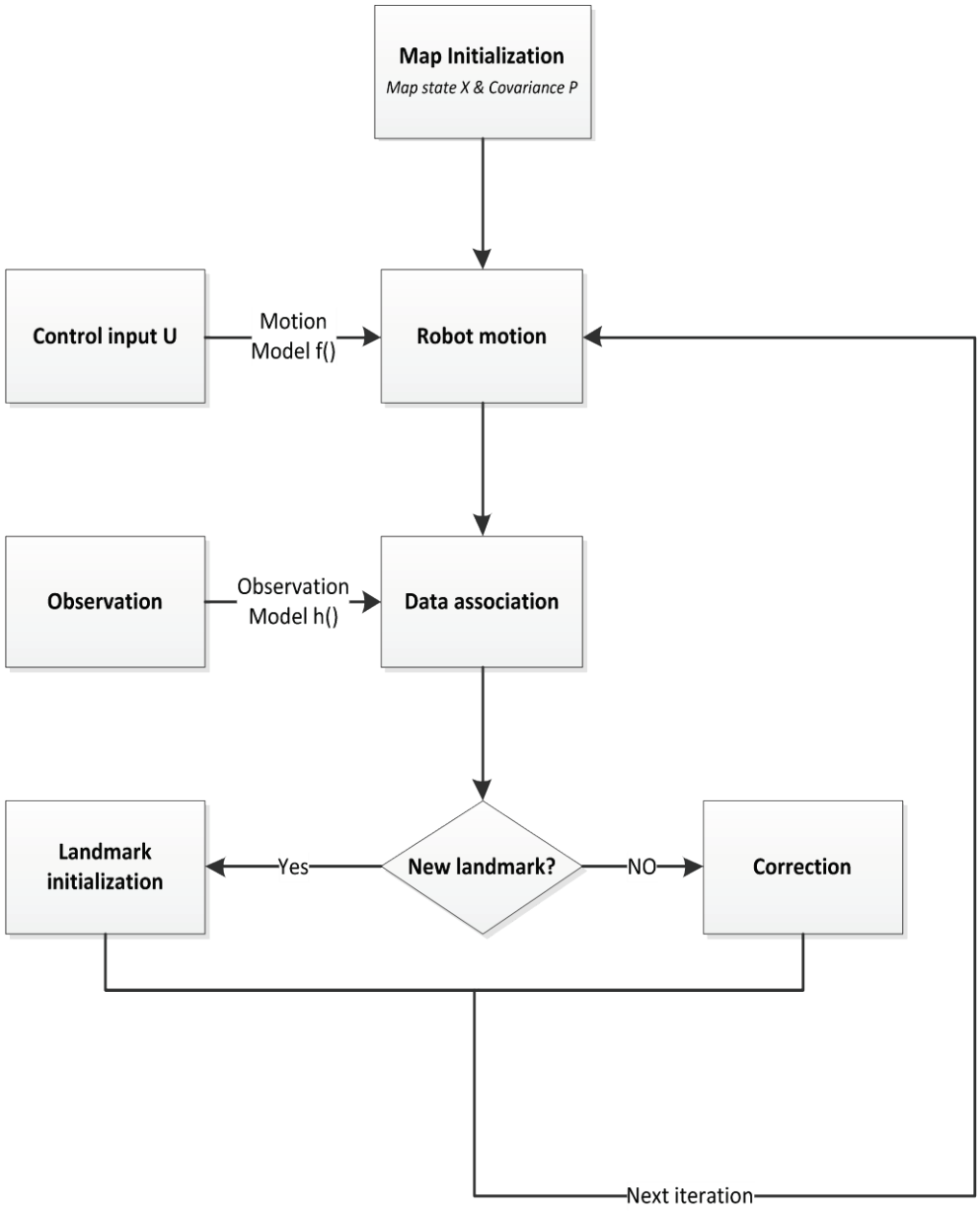


Figure 4.2. The flow chart of EKF-SLAM algorithm

4.2. EKF-SLAM Algorithm Implementation

The implementation of EKF-SLAM algorithm has been realized in both simulation and real-time environments. Considering that the simulated experiment results are obtained through a relatively ideal experimental environment, real-time implementation

on a robot platform is therefore performed to verify the simulated results. The experiments were conducted on the NAO humanoid robot that has been introduced in chapter 3, and the complete EKF-SLAM algorithm was coded and tested in Python programming language compiled by Pydev, a Python IDE (Integrated Development Environment), under Windows environment.

4.2.1. Simulation Experiments & Results

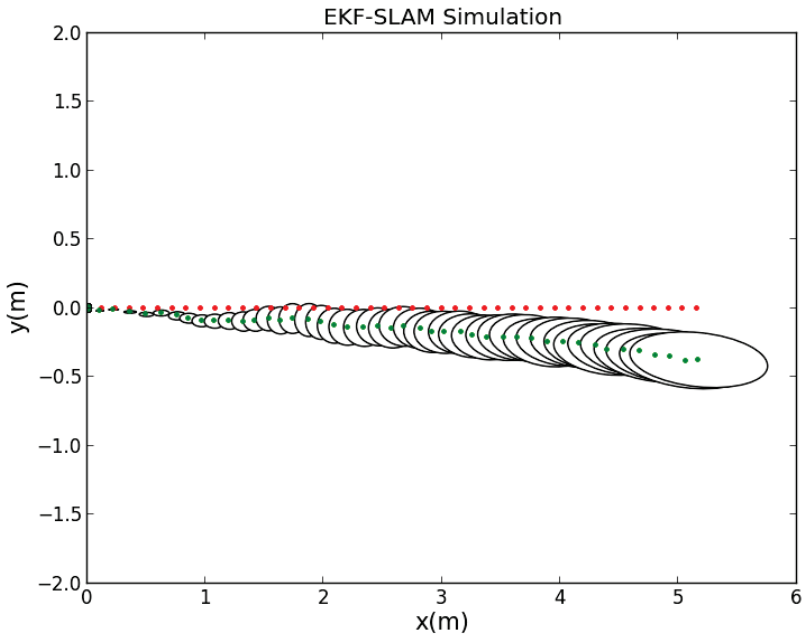
We conducted extensive simulations prior to the real time implementation in order to test the performance of our algorithm in a number of iterations. By running the simulation code, we collected the results on different number of landmarks cases to demonstrate the effect from landmark observation. Additionally, we have calculated the run time for each of the three experiments aiming to show the time complexity.

Case one – No landmark

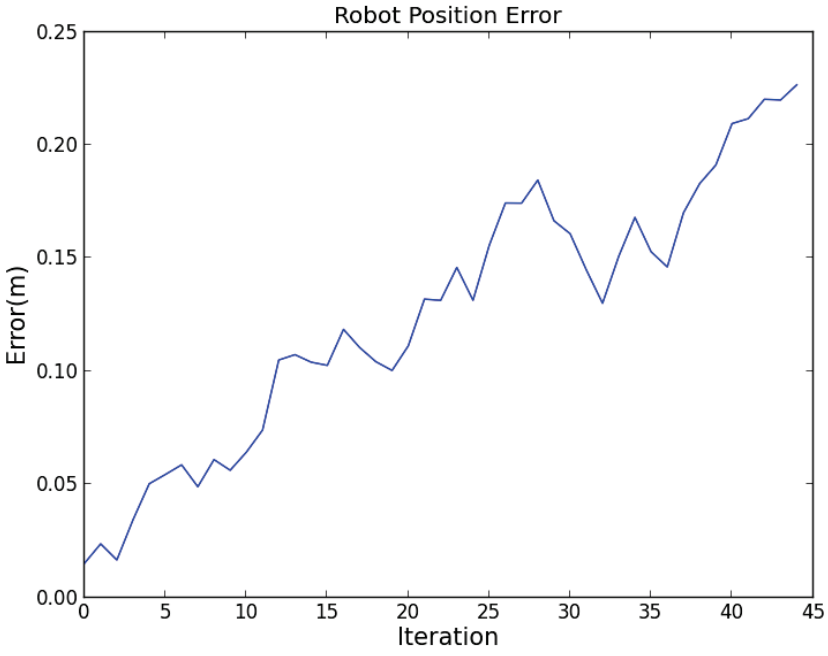
In the first experiment, our purposed EKF-SLAM code is given zero landmark and the control motion of $U = [0.12 \;\; 0 \;\; 0]$, that is robot walking straight with 0.12. The Gaussian noise is given by $SD = [0.02m \quad 0.02m \quad 0.02^r]'$. After running for 45 iterations, the results are illustrated in Figure 4.3(a). We can see that as the robot travels, the estimated position marked with green dots gradually deviate from the reference path in red dot, which is shown as the robot position error in Figure 4.3(b). Similarly, the robot covariance represented in ellipse also diverges and is shown in Figure 4.3(c). Reason for this can be explained by the lack of external data to correct the robot position through the EKF-SLAM algorithm. The output data from the 45th iteration is listed in Table 4.1.

Table 4.1. Experiment result on no landmark case at 45th iteration

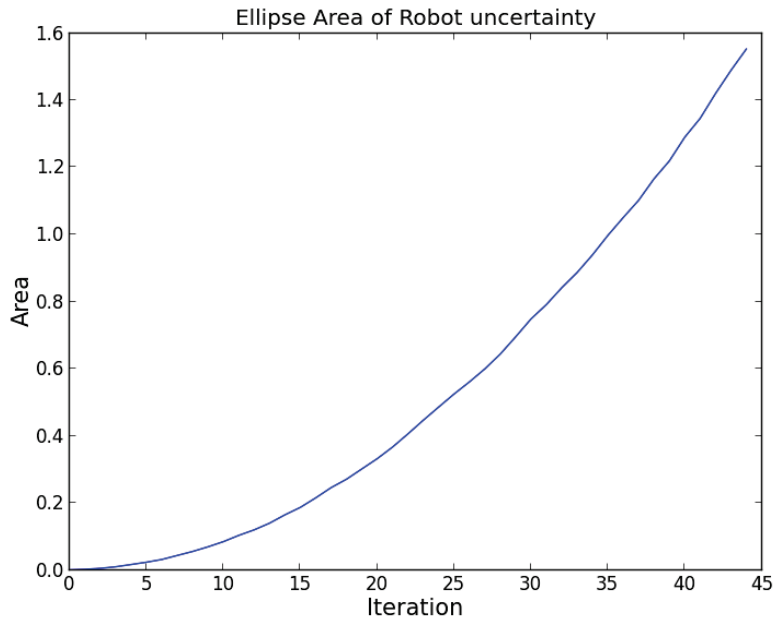
Reference	Estimated	Covariance matrix	Error	Runtime
$\begin{bmatrix} 5.4 \\ 0 \\ 0^r \end{bmatrix}$	$\begin{bmatrix} 5.2423 \\ -0.2165 \\ -0.0595 \end{bmatrix}$	$\begin{bmatrix} 0.0178 & 0.0055 & 0.0018 \\ 0.0055 & 0.1574 & 0.0453 \\ 0.0018 & 0.0453 & 0.0176 \end{bmatrix}$	0.1244m	0.188s


EKF-SLAM Simulation
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y(m)
0
1
2
3
4
5
6
x(m)

(a)


Robot Position Error
0.25
0.20
0.15
0.10
0.05
0.00
Error(m)
0
5
10
15
20
25
30
35
40
45
Iteration

(b)



(c)

Figure 4.3. EKF-SLAM simulation result: a) Plot of estimated robot position denoted in green dots and reference position in red dots. b) the error between estimated and reference robot position. c) the motion uncertainty represented by the area of covariance ellipses grows.

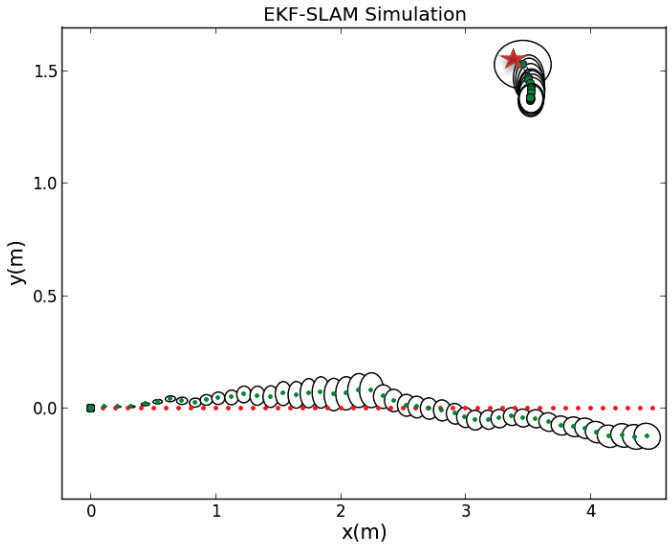
Case two – One landmark

The second simulation experiment demonstrates the landmark observation effect and the result can be compared to the first experiment. The robot used the on-board laser sensor to detect the landmark and to obtain the observation information for the EKF-SLAM algorithm to improve the robot estimate position. Therefore, we added one landmark to the environment at $L = (3.5 \quad 1.5)$ in the global frame. The laser that robot used is set to the range of $2m$ in distance and $(-\pi/2^r, \pi/2^r)$ in bearing. We remain other conditions the same: Gaussian noise for motion and observation are $SD_{motion} = [0.02m \quad 0.02m \quad 0.02^r]'$ and $SD_{obv} = [0.1m \quad \pi/180]'$, control motion $U = [0.12 \ 0 \ 0]$ and running for 45 iterations. The simulation experiment results are depicted in Figure 4.4(a) the red star is the real position of landmark and green dots surrounded with ellipses represent the estimated landmark position and covariance. From the plot, the moment that landmark observation starts to be in effect is easy to identify: nearly after robot passes $2\mathrm{m}$ distance, the estimated robot position approaches to the reference

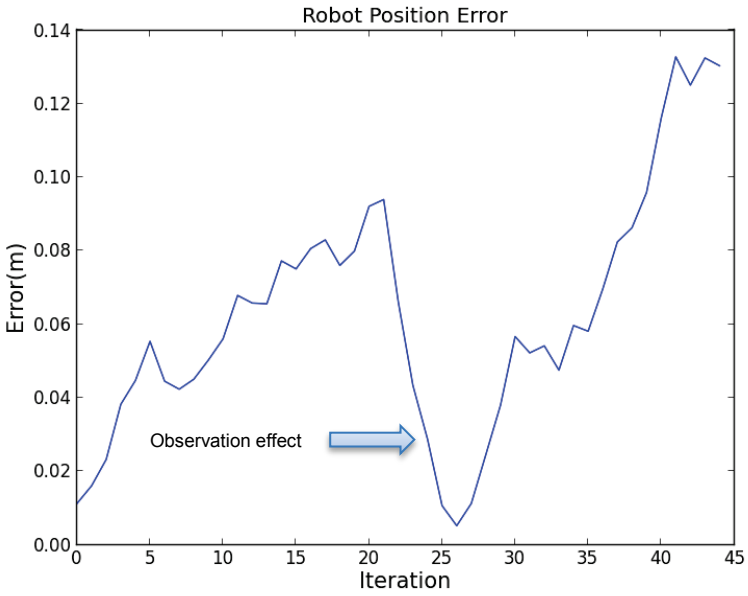
position (Figure 4.4 (b)) and meanwhile the motion uncertainty represented by covariance ellipse drops Figure4.4 (c). Concurrently, the landmark position is being estimated and the data is plotted. Figure 4.5(a) shows that the landmark position error remains certain value after a number of observations. While Figure4.5 (b) points out the decreasing of landmark covariance ellipse begins from the first observation and ends at the last observation in approximately the 33th iteration. In addition, numerical results are provided in the table 4.2**.** Notice that the runtime in this simulation has increased due to the computation expense of landmark observation process.

Table 4.2. Experiment result on one landmark case at 45th iteration

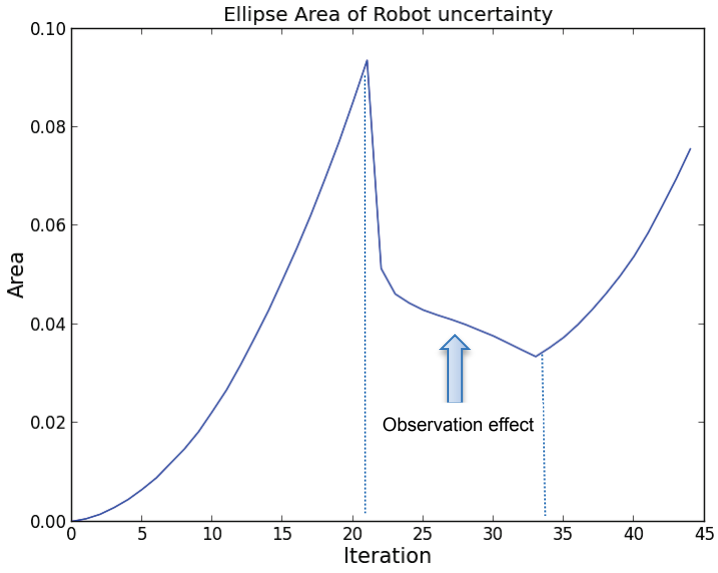
	Ref.	Est.	Covariance matrix	Error	Runtime
Robot	$\begin{bmatrix}5.4\\0\\0^r\end{bmatrix}$	$\begin{bmatrix}5.3488\\0.1399\\-0.0937^r\end{bmatrix}$	$\begin{bmatrix}0.0031 & 0.0017 & 0.0011\\0.0017 & 0.0032 & 0.0018\\0.0011 & 0.0018 & 0.0020\end{bmatrix}$	0.0624m	0.231s
LM	$\begin{bmatrix}3.5\\1.5\end{bmatrix}$	$\begin{bmatrix}3.5215\\1.3793\end{bmatrix}$	$\begin{bmatrix}0.0037 & 0.0051\\0.0051 & 0.0019\end{bmatrix}$	0.0872m	



(a)

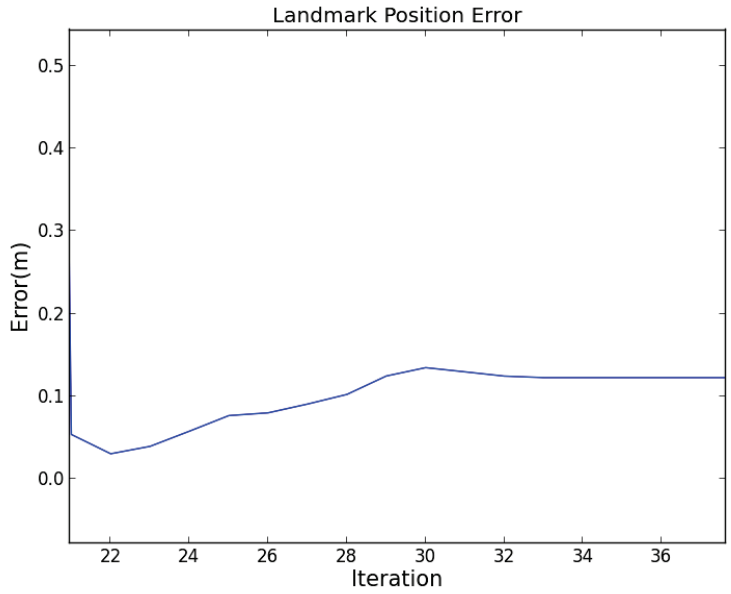


(b)

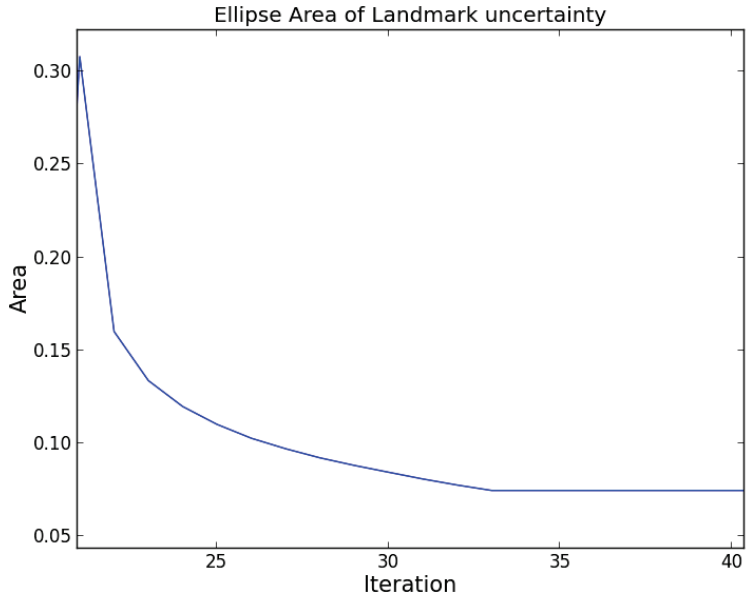


(c)

Figure 4.4. **a) Plot of estimated robot position denoted in green dots and reference position in red dots, with landmark marked in star. b) the drop of error between estimated and reference robot position during observation. c) The motion uncertainty represented by the area of covariance ellipses decreases during landmark observation.**



(a)

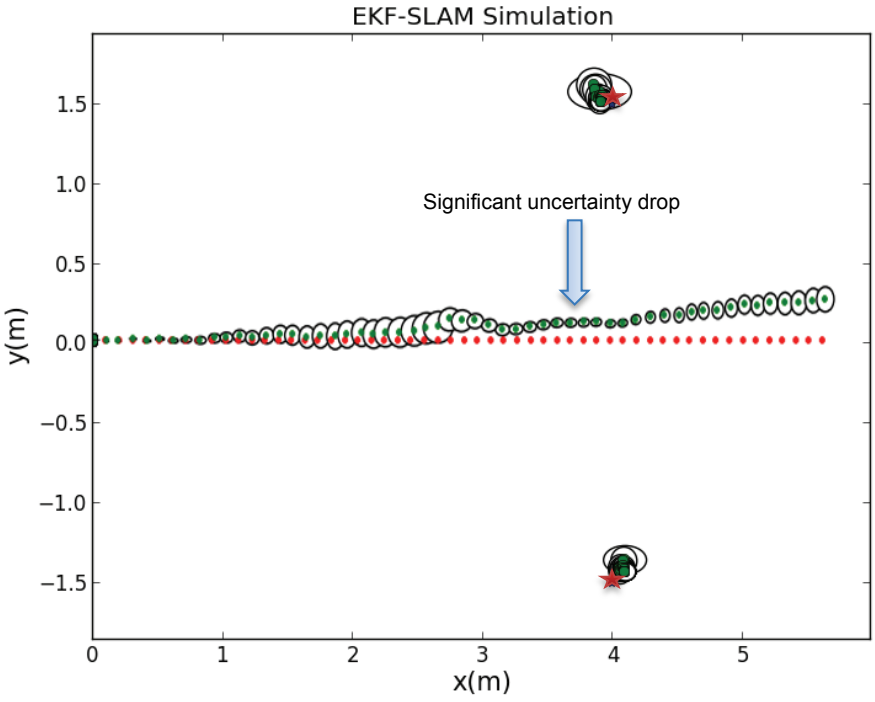


(b)

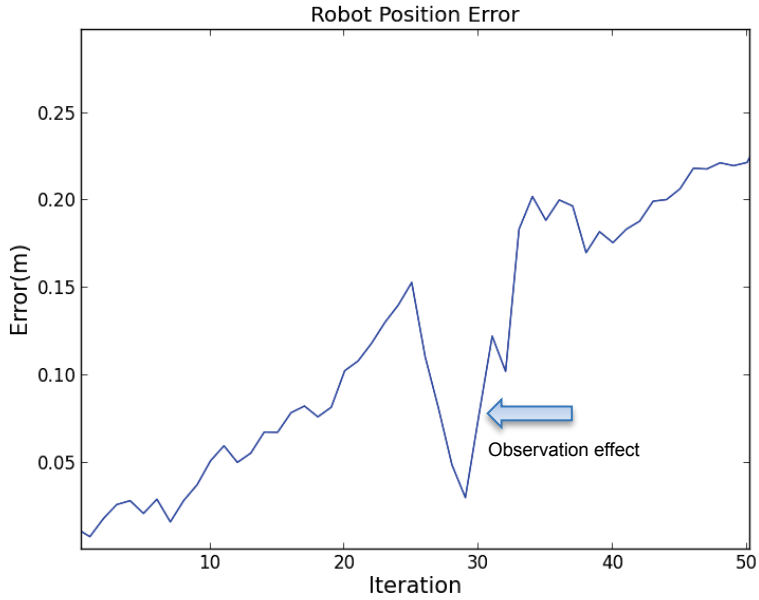
Figure 4.5. Landmark uncertainty: a) landmark position error changes during observation. b) Landmark uncertainty reduces as landmark observation in progress.

Case three – two landmarks

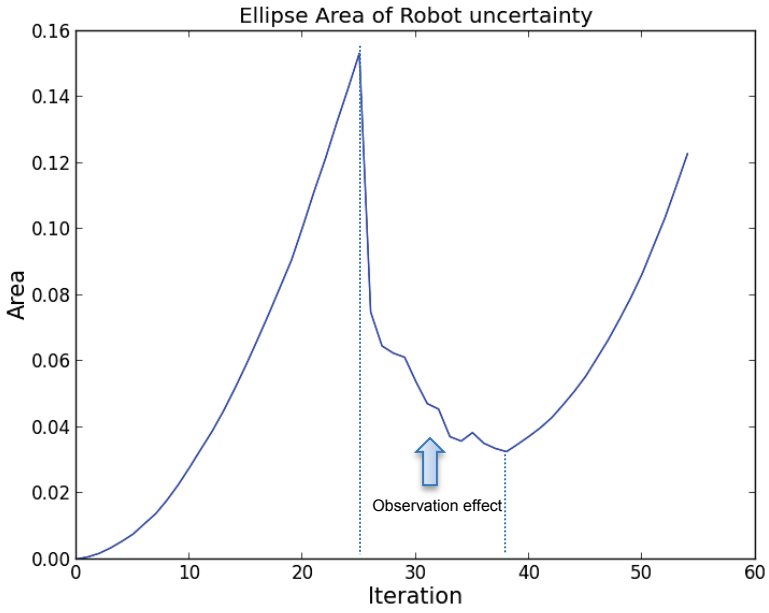
In landmark based EKF-SLAM, one landmark is usually not sufficient to achieve more accurate robot and landmark estimated position. Accordingly, under the same experimental conditions, we introduced a new landmark to the simulation experiment where $L1 = (4 \quad 1.5)$, $L2 = (4 \quad -1.5)$ and obtained a comparative result in Figure 4.6(a). By comparing the two landmarks experiment plot with one landmark plot, one can observe that the uncertainty of the robot motion represented by the ellipse has dropped more significantly while the robot detects multi-landmarks at once Figure 4.6(c). Furthermore, the robot position error decreases during robot observation process, shown in Figure 4.6(b). For other numerical results, see Table 4.3. Note that a longer runtime is required to complete two landmark's simulation.



(a)



(b)



(c)

Figure 4.6. **a) Plot of estimated robot position denoted in green dots and reference position in red dots, with landmark marked in star. b) The drop of error between estimated and reference robot position during observation. c) The motion uncertainty represented by the area of covariance ellipses decreases during landmark observation.**

Table 4.3. Experiment result on two landmark case at 45th iteration

	Ref.	Est.	Covariance Matrix	Error	Runtime
Robot	$\begin{bmatrix} 5.4 \\ 0 \\ 0^r \end{bmatrix}$	$\begin{bmatrix} 5.2384 \\ 0.1493 \\ -0.1293^r \end{bmatrix}$	$\begin{bmatrix} 0.0035 & -0.0016 & -0.0007 \\ -0.0016 & 0.0061 & 0.0033 \\ -0.0007 & 0.0033 & 0.0030 \end{bmatrix}$	0.1231m	0.498s
LM1	$\begin{bmatrix} 4 \\ 1.5 \end{bmatrix}$	$\begin{bmatrix} 4.0999 \\ 1.3672 \end{bmatrix}$	$\begin{bmatrix} 0.0082 & -0.0061 \\ -0.0061 & 0.0118 \end{bmatrix}$	0.0836m	
LM2	$\begin{bmatrix} 4 \\ -1.5 \end{bmatrix}$	$\begin{bmatrix} 4.0547 \\ -1.3986 \end{bmatrix}$	$\begin{bmatrix} 0.0053 & 0.0012 \\ 0.0012 & 0.0048 \end{bmatrix}$	0.0542m	

4.2.2. Implementation of EKF-SLAM algorithm on NAO robot

To verify the simulation results we implemented the EKF-SLAM algorithm onto the humanoid robot NAO and then collected the result from the real time experiment. Models in the EKF-SLAM algorithm are realized using proper NAOqi API models that provided in the Python language version of SDK in NAO software package. Accordingly, previous to the experiments demonstration, we include a brief introduction of NAOqi APIs and discuss their applications in the EKF-SLAM program.

NAOqi APIs introduction and applications in the experiment

Aldebaran has provided a variety of modules for developers to program with NAO and develop advanced applications. These modules also called APIs and can be categorized according to the main function, listed in the table below:

Table 4.4. List of all available NAO APIs

	Description	Modules
Core	Includes modules that are always available in NAOqi.	ALBehaviorManager ALBonjour ALMemory ALModule ALPrerences ALProxy ALResourceManager
Motion	Provides methods which facilitate making NAO move. For example, sending command to walk to specific location.	**ALMotion** ALMotionRecorder
Audio	Manages all functions of NAO audio devices. Commonly used for speaking and voice recognition.	ALAudioDevice ALAudioPlayer ALAudioRecoder ALAudioSourceLocalisation ALSoundDetection ALSpeechRecognition ALTextToSpeech
Vision	Compose of all NAO vision modules. Landmark detection was used for my application	ALFaceDetection **ALLandmarkDetection** ALRedBallDetection ALVideoDevice ALVisionRecognition ALVisionToolBox
Sensors	Deals with NAO sensors that includes infrared, laser sonar and etc.	ALFsr ALInfrared **ALLaser** ALRobotPose ALSensors ALSonar ALLeds
Trackers	This module allows user to make NAO track targets (a red ball or a defined face)	ALFaceTracker ALRedBallTracker
DCM	Stands for Device Communication Manager. In charge of the communication with all electronic devices in the robot except the sound and the camera.	DCM

According to the EKF-SLAM algorithm explained in the previous section, there are two main tasks for NAO robot in the EKF-SLAM process: motion and observation. Each task is completed through related NAOqi API included in the above table.

The motion model in the EKF-SLAM involves the walking motion and positing of the robot, which are achieved from specific method within ALMotion module. The application of ALMotion model firstly takes the control input in *(x, y, theta), where x* and y are the Cartesian coordinates with respect to the robot frame that the robot should approach, with *theta* the final orientation. Furthermore, the end robot position can be then obtained through the odometry data with respect to global frame, used for EKF-SLAM to compute the estimated position.

On the other hand, the observation model in the EKF-SLAM algorithm requires the data from laser sensor on NAO robot. Firstly, ALLaser module permits an access to the configuration of laser such that the laser range in terms of distance and bearing can be customized according to the experiment environment. Then, ALMemory module is called to retrieve the raw data of laser that contains 683 points of data within the coverage of current detection, and each data is composed of 4 parameters describing this point, where first two are the Cartesian coordinates (*x, y*) in the robot frame, and the other two are polar coordinates (*d, ∅)*, where *d* the distance and ∅ the bearing toward the detected point.

Linear motion

This experiment is to realize the EKF-SLAM simulation of the two landmark case on the robot platform. The robot was given a control motion of $U = [0.1 \ 0 \ 0]$, meaning that robot walks straight with $0.1m$ at each iteration. Similar to the simulation experiment, two landmarks were placed in the environment at $L1 = (0.8 \quad 1.5)$, $L2 = (0.8 \quad -0.5)$ in the global frame. The laser range settings were (*20,700mm*) for the distance and ($-3\pi/4, 3\pi/4$) for the bearing. The laser sensor was activated throughout the experiment. Once the landmark enters the laser range, the landmark location in the form of polar coordinate was received and used for the observation step in SLAM. The experiment scenario is shown in Figure 4.7.

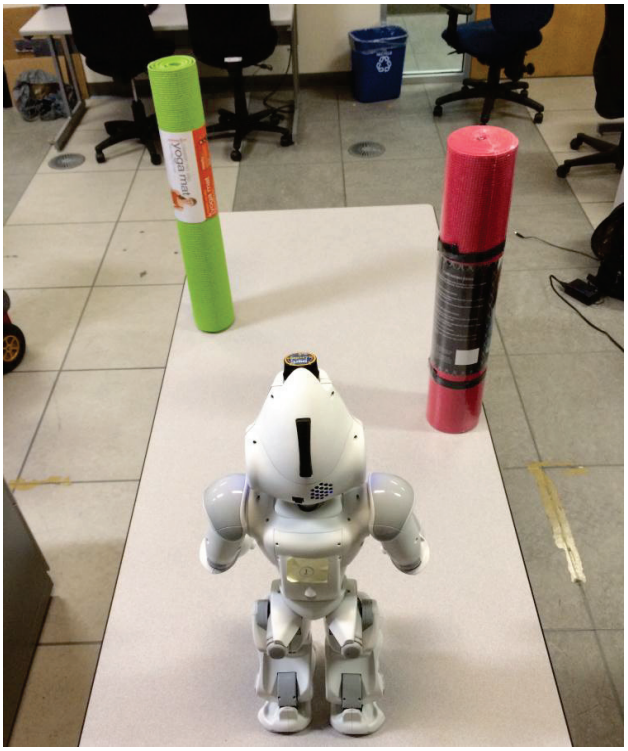


Figure 4.7. EKF-SLAM real-time implementation scenario with two landmarks.

The complete experiment is then plotted in the Figure 4.8. The robot initializes its first step at the initial robot position marked in star, moves straight for 0.1m according to the control motion, obtains the robot odometry data to get the measured position of robot which corresponding to the prediction step in a EKF-SLAM process, and then retrieves laser sensor data regarding the detected landmarks (blue dots surrounding by connivance ellipses) to feed in the EKF-SLAM algorithm in order to minimize the uncertainty of prediction step.

In addition, the existence of deviation is observed during the experiment and is depicted in the plot as well (reference in red and estimated robot position in green separated out after iterations). This deviation is not unexpected as the mechanism of the robot cannot be ensured a perfect symmetry and the ground condition affects the motion deviation. Furthermore, due to the fact that the estimated position in the plot mainly based on solely the odometry data from the robot, which has proven to be not precise, greater amount of deviation between the “real” robot path and reference path is

expected. A technique specifically developed for reducing this deviation will be introduced in the next chapter.

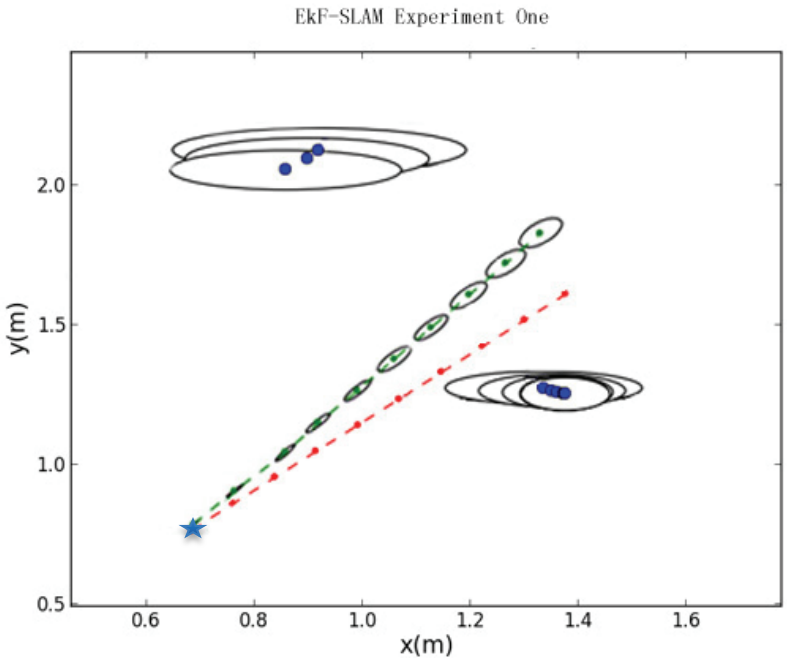


Figure 4.8. Result of real-time EKF-SLAM implementation on two landmark case

Rectangular motion avoiding obstacle

The second experiment is performed in a slightly complicated scenario intending to simulate a practical SLAM exploration mission for a robot. In the experiment, a rectangular shape obstacle was placed in the environment and the robot should be able to avoid the collision while performing the navigation. There are several obstacle avoidance methods available including potential fields, generalized potential fields and vector field histograms [77, 78], while in this experiment a basic avoidance technique based on path planning was implemented. Accordingly, the robot should move along a rectangular path in order to avoid the obstacle, and then retreat to the original location after the observations of totally three landmarks on the path to accomplish the exploration task. The experiment parameters are similar to the linear motion experiment: control motion given by $U = [0.1m \ 0 \ 0]$ at observation, $U = [0 \ 0 \ -\pi]$ for turning and $U = [0.5m \ 0 \ 0]$ for the last step back to origin location, along with laser range settings (*20,700mm*) for the distance and $(-3\pi/4, 3\pi/4)$ for the bearing.

We obtained the result illustrated in Figure 4.9 once experiment is completed. The full experiment is succeeded in 16 iterations of EKF-SLAM algorithm. The runtime is 1m:46.297s. According to the plot, the estimated position of the robot has reached the reference position that is fairly close to where the robot initializes the exploration. However, given that the deviation is currently inevitable without the guidance of external sensor and the robot odometry is not capable of capturing it precisely, a greater deviation is experienced.

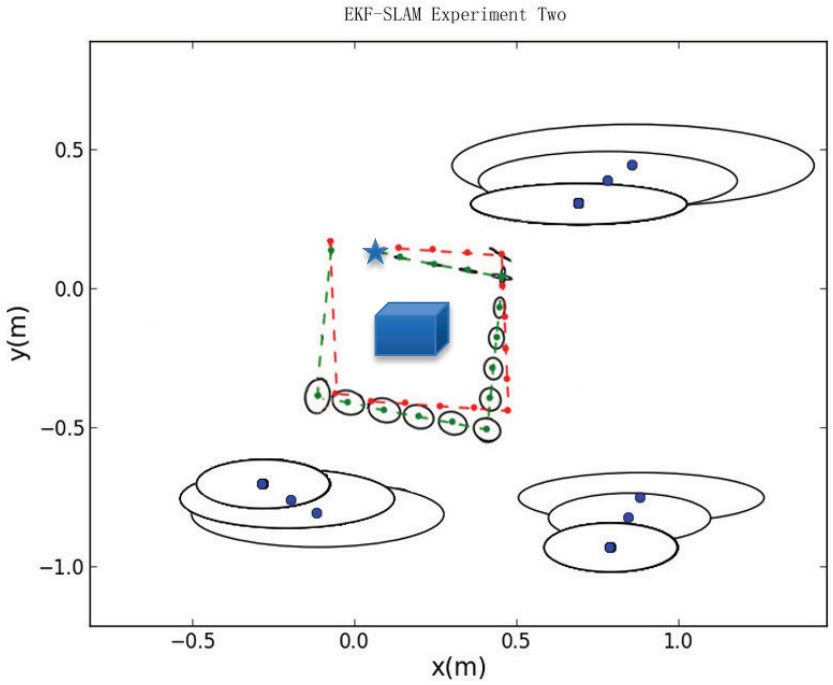


Figure 4.9. Real-time EKF-SLAM implementation result. Robot following a rectangular path to avoid obstacle and retreating to origin.

4.2.3. Summary

In this chapter, a detailed interpretation of EKF-SLAM algorithm is firstly presented aiming to provide readers a comprehensive knowledge of EKF-SLAM algorithm, which is considered as the foundation technique of my thesis project. The next portion of this chapter discusses the EKF-SLAM on simulation and experimental implementation on humanoid robot NAO. The simulation results have validated the effect of observation model where the uncertainty of robot motion minimized during the landmark observation process, and also have verified that the time complexity of the

algorithm is increased with the number of landmark to be detected by the robot. On the other hand, the full real-time implementation of EKF-SLAM on robot NAO succeeded in two scenarios. The first scenario is the realization of two landmark simulation, and the second experiment is design particularly to simulated a practical exploration task in an unknown environment to a robot platform, where the NAO robot traveled along a assigned rectangular shape path in order to accomplish the EKF-SLAM process and meanwhile avoiding an obstacle and returning to where it starts. The results for both experiments have proven the effectiveness of the implementation of EKF-SLAM, while one experimental issue, the motion deviation is observed and will be studied in the next chapter.

Chapter 5.

Augmented Reality Implementation for Robot Navigation

The objective of Augmented Reality technology, as stated in the literature review chapter, is to enhance the information acquisition of practical world by augmenting with computer-generated sensory input. In this chapter, we demonstrate extended studies on the previous research of landmark based EKF-SLAM by implementation Augmented Reality. Two distinctive applications of Augmented Reality have been integrated with the original EKF-SLAM program for performance improvement. The first application involves the vision recognition of specific landmarks for obtaining predefined landmark information, such that the robot obtains the capability of navigating in a dense environment with multiple landmarks and obstacle. The landmark information can also be used for the simplification of data association problem in EKF-SLAM. The second application is aiming to enhance the precision of existing robot odometry and therefore reduce the deviation from robot walking by taking advantage of the external data from iPhone gyrometer to update the odometry and a implementing a PI motion controller for position correction.

5.1. Vision recognition augmented EKF-SLAM implementation on NAO robot

In the first phase of Augmented Reality improved EKF-SLAM experiment, we applied the landmark recognition function from NAO robot API package in order for the robot to recognize landmarks that are attached with NAOmarks and obtain pre-loaded information useful for navigation. The Module of landmark recognition is therefore reviewed in the following subsections.

5.1.1. Landmark Recognition on NAO

Object recognition problem has been broadly studied in the areas of computer vision and image processing, which dealing with finding and identifying objects in an image or video sequence [79]. One can sense the significance of this problem to an Augmented Reality application that before augmenting with additional information, the particular object in the physical world has to be recognized by the system at the first place. In order to achieve the object recognition task on NAO robot, the landmark detection module provided in NAOqi APIs is used.

The Landmark detection module enables NAO to recognize special landmarks with specific patterns called NAOmarks. NAOmarks are logos consist of black circles with white triangle fans centered at the circle's center. The landmark recognition module can identify the particular location of the different triangle fans and return their NAOmark ID which is two to three digital numbers [80]. Figure 5.1 shows the sample NAOmarks that used in the experiment.

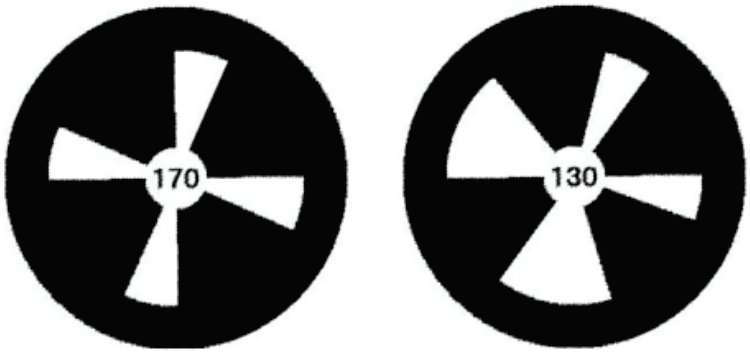


Figure 5.1. NAOmarks with mark ID in the center [80]

NAOmark detection is achieved by applying ALLandmarkDetection module in NAOqi APIs. Main steps regarding the application of this module presented in the form of Python programming language is listed in the following table.

Table 5.1. NAOmark detection steps

	Python code	Description
1	markProxy = ALproxy("ALLandMarkDetection", IP, PORT) memProxy = ALproxy("ALMemory", IP, PORT)	Creating proxy to NAOmark detection module and NAO's memory module
2	period = 500	How often to detect NAOmark and output results (in milliseconds)
3	markProxy.subscribe("Test_Mark", period, 0.0)	Subscribing to NAOmark detection extractor
4	Result = memProxy.getData("LandmarkDetected")	Get result from NAOmark detection

The obtained results mainly consist of shape information and extra information of detected NAOmarks:

- *ShapeInfo = [0, alpha, beta, sizeX, sizeY, heading]* . alpha and beta represent the Naomark's location in terms of camera angles ; sizeX and sizeY are the mark's size in camera angles ; the heading angle describes how the NAOmark is oriented about the vertical axis with regards to NAO's head.
- *ExtraInfo = [MarkID]* . Mark ID is the number written on the NAOmark and which corresponds to its pattern. This Mark ID is used in the project to assist robot distinguish different landmarks.

5.1.2. Experimental implementation and results

The integration of Augmented Reality with the EKF-SLAM algorithm is succeeded on NAO robot based on the use of NAOmark recognition function. Generally, the fundamental structure of EKF-SLAM remains while the Augmented Reality processes take place when NAOmark is detected, where additional information is retrieved regarding to the detected NAOmark and next NAOmark to go. The main contribution of integrating Augmented Reality into EKF-SLAM is the use of this additional information to assist robot in navigation task within a practical environment.

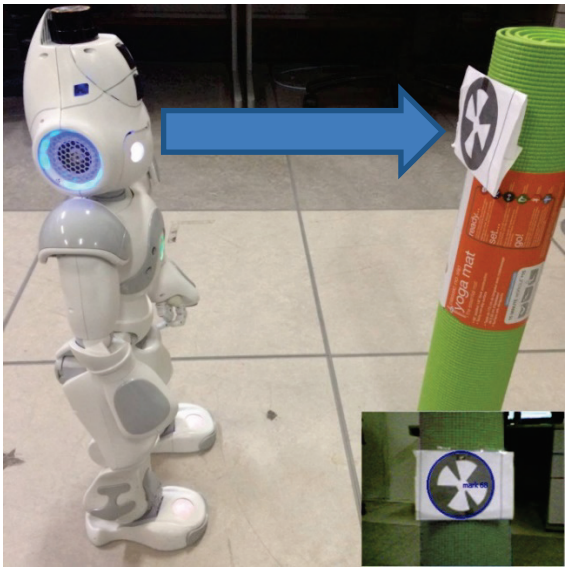


Figure 5.2. NAO detecting NAOmark and output the Mark ID

Augmented Reality implementation

Initially, the EKF-SLAM algorithm performs regular map initialization process and then starts its regular motion function. Differently, before the EKF-SLAM conducts the observation function using laser sensor, NAO calls the landmark detection module aiming to find if there are NAOmarks in the current camera field of view. This process is shown in Figure 5.2, NAO stops in front of landmark attached with NAOmark, landmark detection module should report the detected NAOmark by circling them and MarkID displayed next to it. Once the right NAOmark is detected and Mark ID is retrieved, extra information can be achieved corresponding to the Mark ID number, which is inspired by Augmented Reality. Two pieces of predefined information are received from NAOmark detection, which includes:

- The control motion U to the next landmark:

  Given by $U = [x\ y\ \theta]$. The control motion should lead the robot to the next landmark to be detected. Accordingly, x in the control motion is assigned by an adjusted walk distance for the robot to reach the detection of next landmark. By adjusted, it means that the robot should be able to be in the range of detecting next NAOmark according to the walk distance. This distance is derived from distance formula:

$$\mathbf{d} = \sqrt{(\boldsymbol{x_{est}} - \boldsymbol{x_{lm}})^2 + (\boldsymbol{y_{est}} - \boldsymbol{y_{lm}})^2} \qquad \textbf{Equation 5.1}$$

Where $x_{est}\ and\ y_{est}$ the current estimated robot position, $x_{lm}\ and\ y_{lm}$ next landmark position according to information from NAOmark. Based on several experiments, we use $\mathrm{x} = 3\mathrm{d}/4$ to obtain a proper distance between NAO robot and NAOmark as illustrated in Figure 5.2

On the other hand, the turning angle θ toward next landmark is calculated from:

$$\boldsymbol{\theta} = \mathbf{tan^{-1}} \frac{(\boldsymbol{y_{lm} - y_{est}})}{(\boldsymbol{x_{lm} - x_{est}})} \qquad \textbf{Equation 5.2}$$

- The Mark ID:

  Once the Mark ID is extracted, we append the map state with one dimension to store the Mark ID number as identifier:

$$L = \begin{bmatrix} x_{lm1} \\ y_{lm1} \\ x_{lm2} \\ y_{lm2} \\ \dots \end{bmatrix} \quad \Rightarrow \quad L = \begin{bmatrix} x_{lm1} \\ y_{lm1} \\ \mathrm{id}_1 \\ x_{lm2} \\ y_{lm2} \\ \mathrm{id}_2 \\ \dots \end{bmatrix} \qquad \textbf{Equation 5.3}$$

  This process simplifies the data association problem because that the corresponding landmark can be easily matched once the Mark ID is re-observed, without having to compute Mahalanobis distance frequently.

In summary, the procedure of the Augmented Reality part is described in steps as follow:

1. Robot motion according to NAOmark indication
2. Calling landmark detection module to identify NAOmark
3. Retrieve pre-loaded landmark information corresponding to which NAOmark is detected
4. Continue regular EKF-SLAM

Figure 5.3. AR-EKF-SLAM experiment scenario. NAO walks to and observes landmark one by one and return to original location

Full Experiment Demonstration

The experiment scenario has taken the idea of rectangle motion avoiding obstacle experiment in EKF-SLAM. Three landmarks located at corners of rectangle with an obstacle placed in the center. Differently, three landmarks all attached with NAOmarks that printed on pieces of papers. The control motion is variant according to the next landmark location, along with laser range settings (*20,700mm*) for the distance and $(-3\pi/4, 3\pi/4)$ for the bearing. NAO robot should finish the task of exploring the experiment environment in Figure 5.3 using EKF-SLAM algorithm following the path directed by NAOmarks. The full experiment is depicted in Figure 5.5.

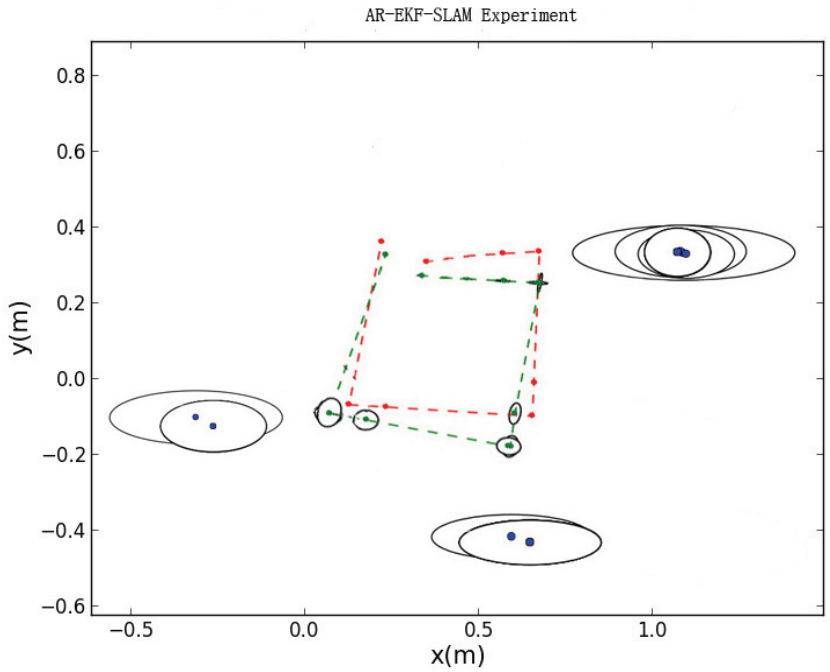
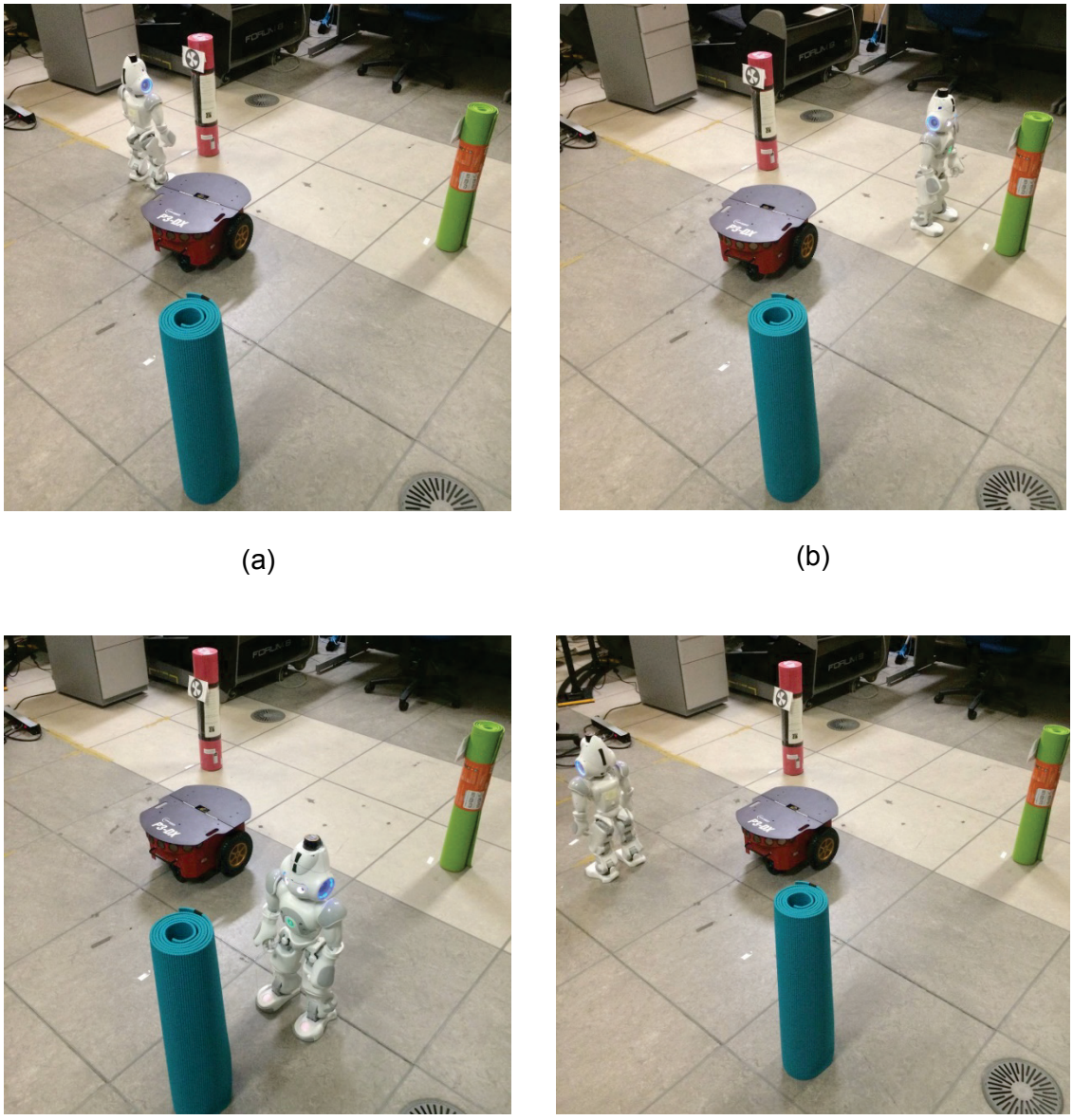


Figure 5.4. Result of AR-EKF-SLAM experiment. Slight deviation can be observed

We then plotted the result in Figure 5.4, where the tasks of observations to three landmarks and the robot motion to origin are succeeded in 11 loops with the spend of 1m:18.899s in time. Nevertheless, based on the experiment observation, NAO robot motion contained greater deviation than it appeared in the plot, due to the limited accuracy of robot odometry. In summation, Figure 5.6 presents a flow chart of the overview of AR-EKF-SLAM algorithm introduced in this section.



(a) (b)

(c) (d)

Figure 5.5. AR-EKF-SLAM experiment: a) NAO stops in front of first NAOmark with a proper distance start NAOmark recognition and landmark detection. b) NAO makes its move and reaches the second landmark. c) NAO arrives at the last landmark, EKF-SLAM completed. d) NAO retreats at original location.

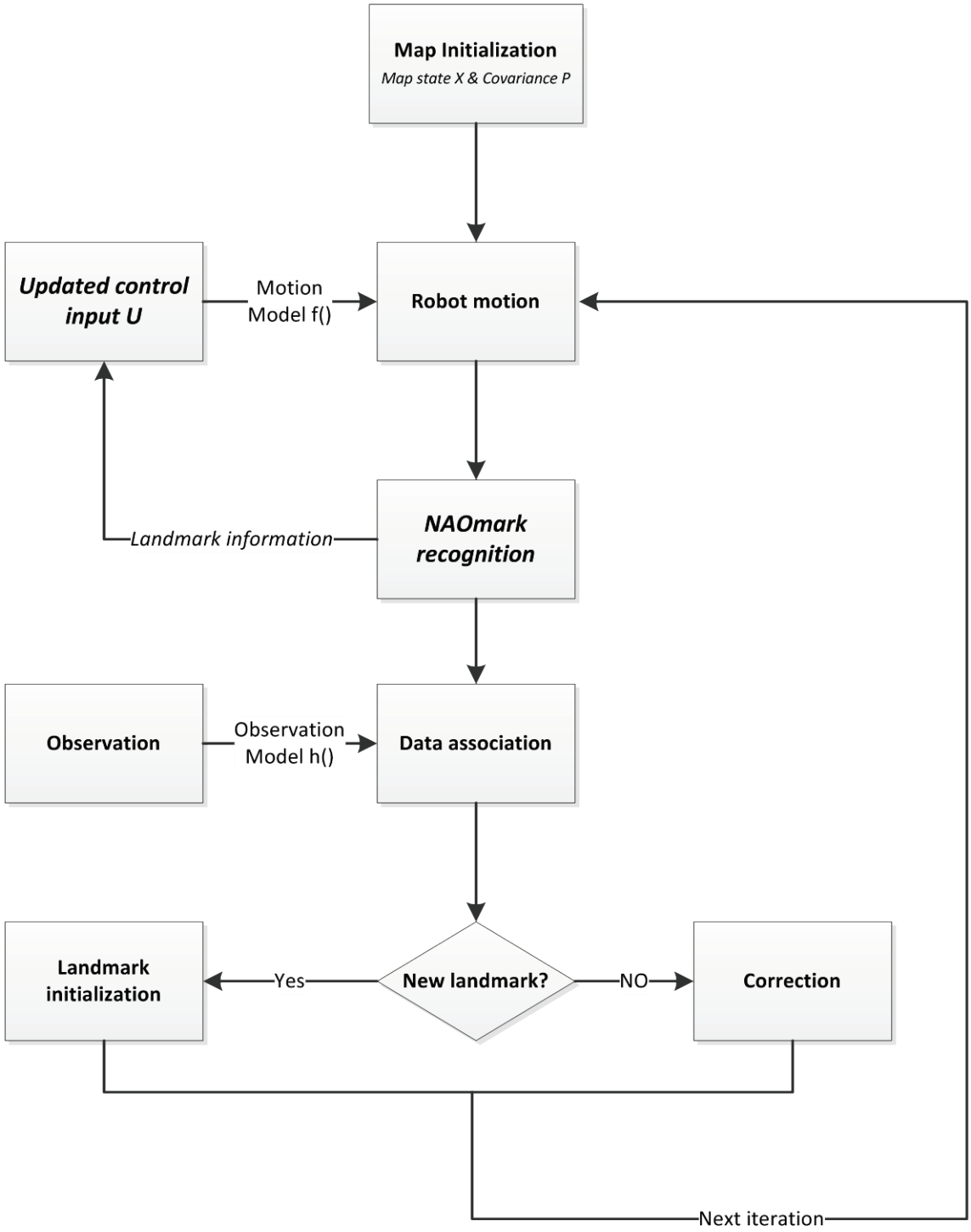


Figure 5.6. Overview of AR-EKF-SLAM algorithm

5.2. Reduce NAO robot position error using iPhone gyrometer with closed-loop controller

In this section, we study on a solution to the deviation problem of NAO robot. Firstly, we managed to replace the use of robot odometry and augmented the odometry with iPhone gyrometer sensor data to obtain a more accurate robot position. Then, this improved robot position is used for a simple closed-loop PI controller to reduce the error between reference position and estimated position. The performance of this method is tested with EKF-SLAM two landmarks linear motion experiment then extended to the full AR-EKF-SLAM method.

5.2.1. Problem description

During the experiments, it was observed that the trajectory of the robot deviates from the planned trajectory all the time. The cause for two legged robot motion deviation can be varied. For example, each leg may step differently while walking due to the control complexity of leg mechanism. The ground condition may also reflect the robot motion. Interestingly, not only this problem exists in robotics, it happens to human as well: suppose that one asked to walk strictly straight with eyes covered, after certain distance, the tester will always deviates from the straight path, and this error between actual and planned position accumulates as movement continues. Therefore, in order to reduce this position error, the tester needs to use one or some of his senses, vision for instance, to make observation and correction according to the amount of derivation.

Similarly, this idea can be adopted into robotics. The data from robot odometry should indicate the position error, and then a proper motion controller is integrated to guide the robot back to the reference path. Whereas, when it comes to our NAO platform, it has already been mentioned previously that the actual deviation observed from experiment is usually greater than what it appears from the plot of odometry data. We then found that the odometry that NAO robot relies on is based on dead reckoning in which error accumulates as robot moves. Therefore, we need a replacement of more accurate positioning system using external devices such that the motion controller is able to make correction based on the actual deviation.

5.2.2. *Odometry Improvement*

In order to obtain a precise odometry data, the NAO robot platform was augmented with an external gyroscope sensor equipped on the most popular smartphone iPhone. The gyroscope detects any change in an object's axis rotation, including pitch, yaw and roll with a desired precision. Assume that iPhone is lying down on a plane, the distribution of pitch yaw and roll are shown in Figure 5.7. Accordingly, if placing the iPhone on NAO robot with same pose, the yaw data should represent the orientation of NAO. The placement of iPhone with NAO robot is depicted in Figure 5.8.

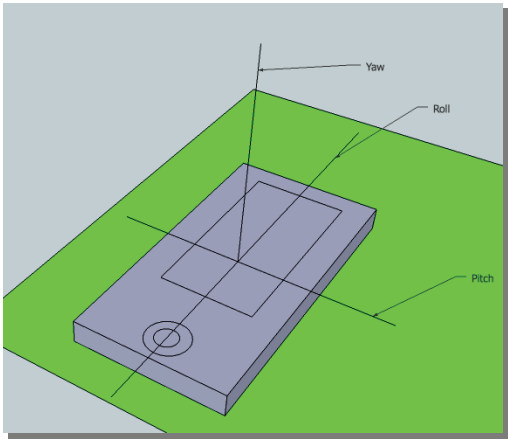


Figure 5.7. Pitch, roll and yaw on an iPhone

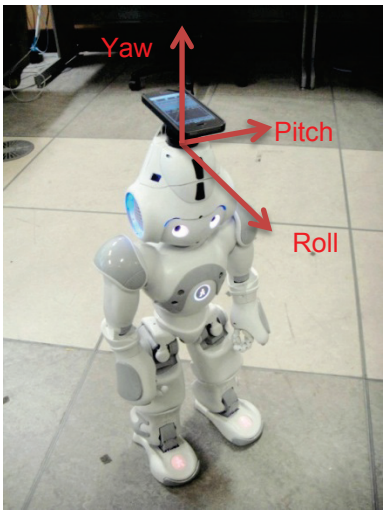


Figure 5.8. NAO robot mounted with iPhone to receive gyrometer data

There are a number of applications available for iOS devices to gain access to on-board sensor data. After testing and comparing these applications, SensorLog Version1.4 has won the competition from its convenience and precision. This application sends online stream sensors data which is of great importance to the need of our proposed implementation. Figure 5.9 shows the interface of SensorLog application outputting gyrometer data. To receive the data, a paragraph of Python code that provides the access a tcp/ip connection on a dedicated socket port was prepared. The IP address and socket port used in the experiment are 192.168.1.66 and 64646, respectively, and the data streaming rate was set to 500ms according to several tests.

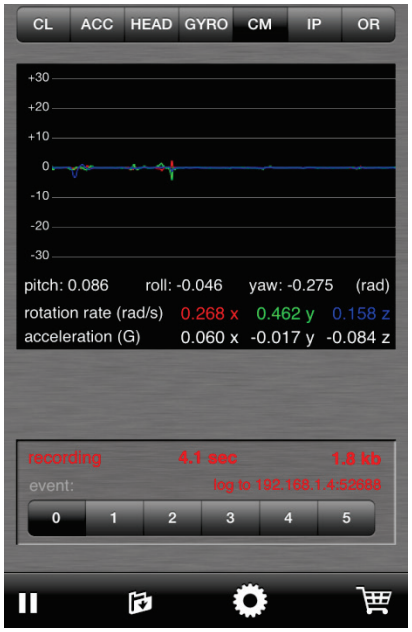


Figure 5.9. Interface of SensorLog iPhone application (https://itunes.apple.com/ca/app/sensorlog/id388014573?mt=8)

Once the yaw data which also representing the robot orientation is filtered from other unused data, we store it as the actual orientation of end robot position, denoted as θ_{yaw}. The motion model f () is used to calculate the x and y of end robot position. Thus, the new enhanced robot positioning system can be derived by:

$$\boldsymbol{R_t = f\,(R_{t-1}, u_t, n_t)} = \begin{pmatrix} \boldsymbol{x_r} \\ \boldsymbol{y_r} \\ \boldsymbol{\theta_r} \end{pmatrix} \qquad \textbf{Equation 5.4}$$

Where R_t and R_{t-1} the post and last robot positon, u_t the control motion, n_t noise vector.

$$\boldsymbol{R_t} = \begin{pmatrix} \boldsymbol{x_r} \\ \boldsymbol{y_r} \\ \boldsymbol{\theta_r} \end{pmatrix} \rightarrow \begin{pmatrix} \boldsymbol{x_r} \\ \boldsymbol{y_r} \\ \boldsymbol{\theta_{yaw}} \end{pmatrix} \qquad \textbf{Equation 5.5}$$

In this step, the estimated robot orientation θ_r is replaced by θ_{yaw} the actual orientation obtained from iPhone gyrometer.

5.2.3. PI Motion Controller

An enhanced positioning system proposed in the last subsection outputs estimated position data that closer to the actual robot position than using NAO's odometry data. Subsequently, the error between estimated and reference robot position can be calculated, which is in significant amount during the experiment. Accordingly, a closed-loop PI controller is implemented to minimize this error, such that the deviation that robot travels can be reduced. This PI controller is mainly based on the return error of estimated and reference position error and is outlined in Figure 5.10.

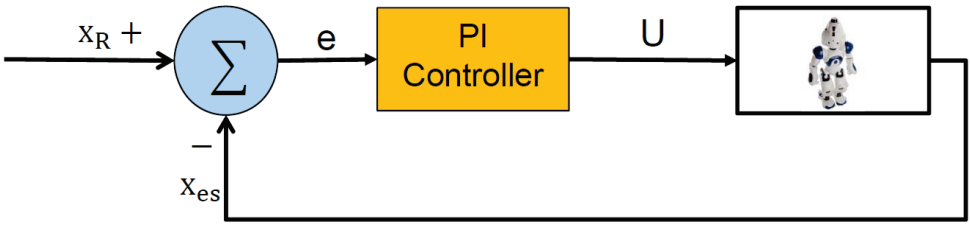


Figure 5.10. The closed-loop motion controller used in the project [11]

Where x_R is the reference position, x_{es} is the estimated position.

The Closed-loop controller steps are:

1. Initialize controller value as a constant: $\mathrm{U} = \mathrm{Constant}(\mathrm{i.e.\,U} = 0.1)$
2. ***if*** the error $\mathrm{e} = \mathrm{x_R} - \mathrm{x_{es}} > \mathcal{E}$ ($\mathcal{E}$ is the threshold defined by user), then $\mathrm{U} = \mathrm{k_p}(\mathrm{e} + \mathrm{T}/\mathrm{T_i} \sum \mathrm{e})$, ($\mathrm{k_p}$ the proportional value)

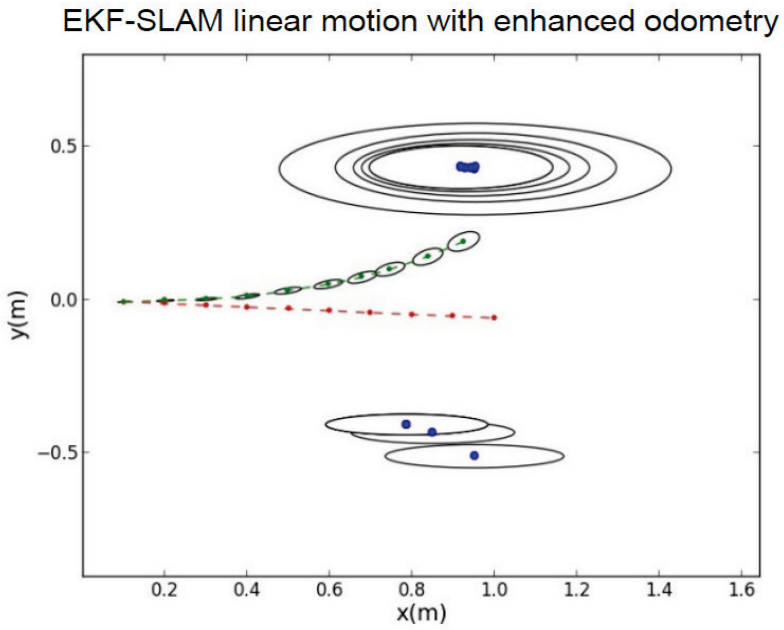
   else $\mathrm{U} = \mathrm{Constant}\ (\mathrm{i.e.\,U} = 0.1)$

5.2.4. Experimental implementation and results

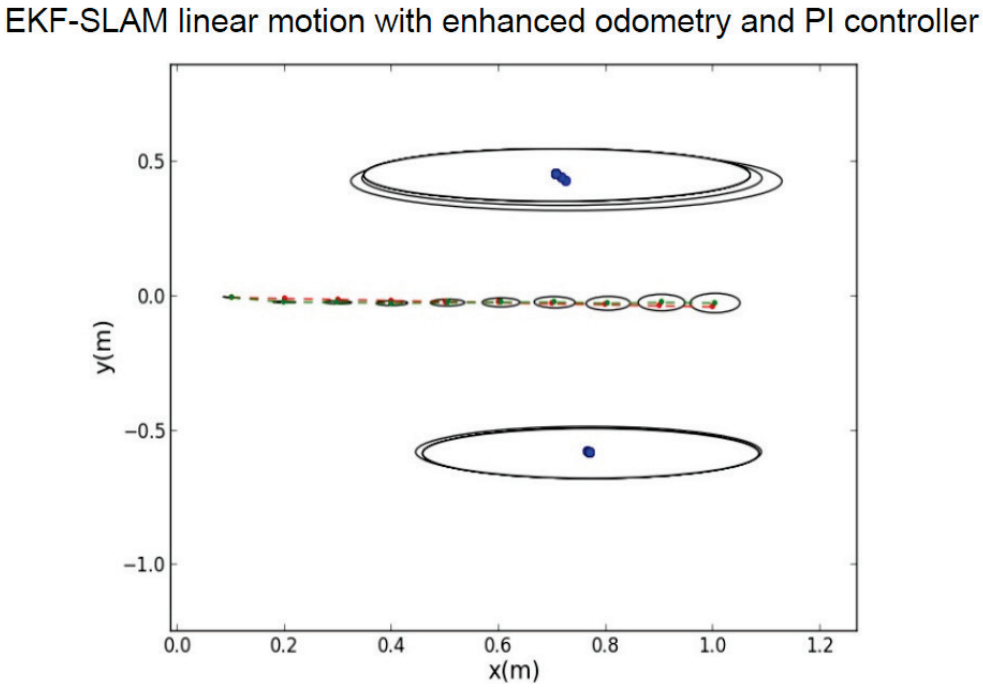
The enhanced robot positioning system and motion controller was integrated with the EKF-SLAM and AR-EKF-SLAM in order to enhance the performance of these algorithms. In the first group of experiment, the robot was asked to perform an AR-EKF-SLAM navigation following a straight line path, with a distance of 0.1m for every calculation iteration and with two landmarks. Figure 5.11(a) presents the experiment result of using new enhanced odometry on EKF-SLAM. We can observe that the estimated robot position marked in green dot deviates from the reference path in red dash line at each iteration, which clarifies the accumulated position error as robot moves forward. With adding the motion PI controller, the result is depicted in Figure 5.11(b). From the figure we can see that the estimated position has remained close to reference path, which verifies the effectiveness of motion controller

Furthermore, this method has been test in a more complicated experiment environment based on AR-EKF-SLAM algorithm testing scenario. Results are recoded and plotted. Figure 5.12(a) presents the result using the proposed positioning system. In the estimation, deviation has been captured by the odometry at each observation iteration. According to the plot, larger amount of deviation occurred at turning points on the robot path, and this corresponds to the past experience with the robot, that the amount of turning angle cannot ensure to be very precise due to the mechanical structure of robot leg. As a result, the robot completed the navigation at somewhere away from the prescribed location. In contrast, the implementation of motion controller has proven its ability of correction according to the result plotted in Figure 5.12(b), where the robot has shown to follow the reference path nicely and nearly return to the initial location.

In addition, to verify the correctness of the enhanced robot odometry and the performance of motion controller, each result has been verified with observation during the experiment. It has shown that the new odometry is capable of representing the actual position of robot, and the motion controller has succeeded in keeping NAO robot with small position error.

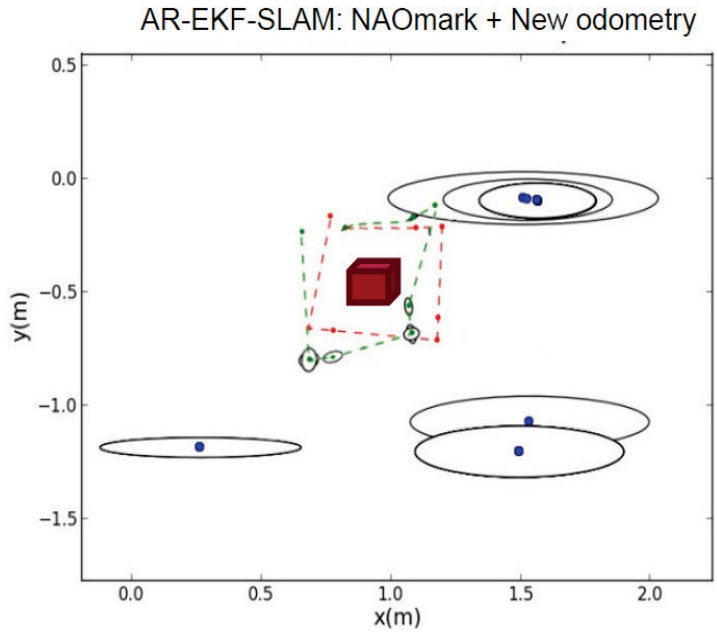


(a)

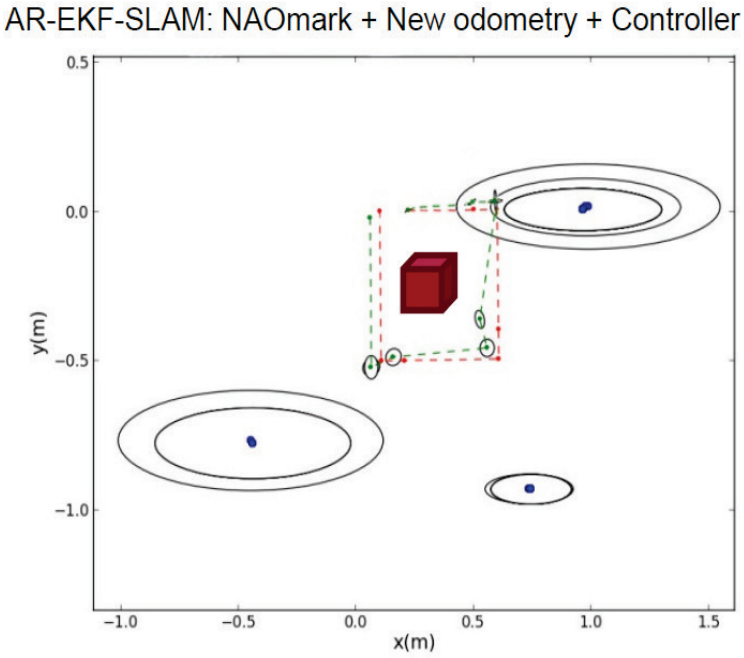


(b)

Figure 5.11. EKF-SLAM experiment two landmark results: a) with improved odometry b) with both improved odometry and controller



(a)



(b)

Figure 5.12. AR-EKF-SLAM full experiment: a) with improved odometry b) with both improved odometry and controller

5.3. Summary

This chapter provides a comprehensive presentation of the integration of Augmented Reality with EKF-SLAM algorithm. We managed to develop two applications of AR and use it to improve the performance of EKF-SLAM. In the first application, NAOmarks that can be recognized by NAO's camera were pre-loaded with augmented information such that NAO robot was able to accomplish the navigation task in a dense environment with multiple landmarks and obstacle. This information was also used in data association to provide a clear identifier of which landmark it is. The second application, iPhone on-board gyrometer data was extracted and received by a dedicated program and then used in EKF-SLAM's motion module to obtain a more precise estimation of robot position. This estimated position is then used to calculate the position error and is fed into a closed-loop motion controller. Experiments have verified the correctness of enhanced robot positioning system. In the next chapter, we aim to demonstrate the contribution of implementing Augmented Reality by comparing and analysing the experimental results of EKF-SLAM and AR-EKF-SLAM.

Chapter 6.

A Comparison of EKF-SLAM and AR-EKF-SLAM

In this chapter, the results from the original EKF-SLAM experiment and from the two applications of Augmented Reality are presented and compared. Explanations are provided on the difference of these results, which indicate the improvement that has been accomplished based on the original EKF-SLAM algorithm. We collected three sets of results: the average error, which is the average position error between the estimated and reference robot position throughout the experiment; the iterations, indicating the total iterations of EKF-SLAM spend to complete the experiment; and the runtimes, the overall time spend to finish the experiment.

6.1. Result comparison and study of linear motion experiment

We applied the iPhone gyrometer enhanced NAO odometry system onto the EKF-SLAM experiment with two landmarks and linear motion scenario. The results plotted in the Figure 6.1(b) where the reference path and measured position of robot are indicated as dash line connected red dots and green dots, respectively. The blue dots represent the observed estimated landmark location and the ellipses surrounding measured position and estimated landmark position descript the uncertainty. From the plot one can observe that the estimated robot path in green dots deviates from the reference path very small at the beginning and increases as robot moves forward, considering as a curved line. This can be explained by the fact the deviation should exist at each robot motion iteration, resulting in the accumulated deviation. In contrast, Figure 6.1(a) shows a different result from the same experiment while using the NAO on-board odometry. The robot estimated path in green dot follows a straight line, which indicates that the robot deviation is doubtfully only existed at the origin and then travels in a

straight line. By comparing the result with the actual observation during the experiment, it is concluded that the result from the iPhone sensor enhanced odometry in Figure 6.1(b) represents the true robot motion path more precisely than the robot on-board odometry. Furthermore, with the implementation of PI motion controller, the robot has also been observed to walk nearly along a straight line that close to the reference path, resulted displayed in Figure 6.1(c). In summary, selected results are list in Table 6.1, in which the average error with enhance odometry and controller largely reduced.

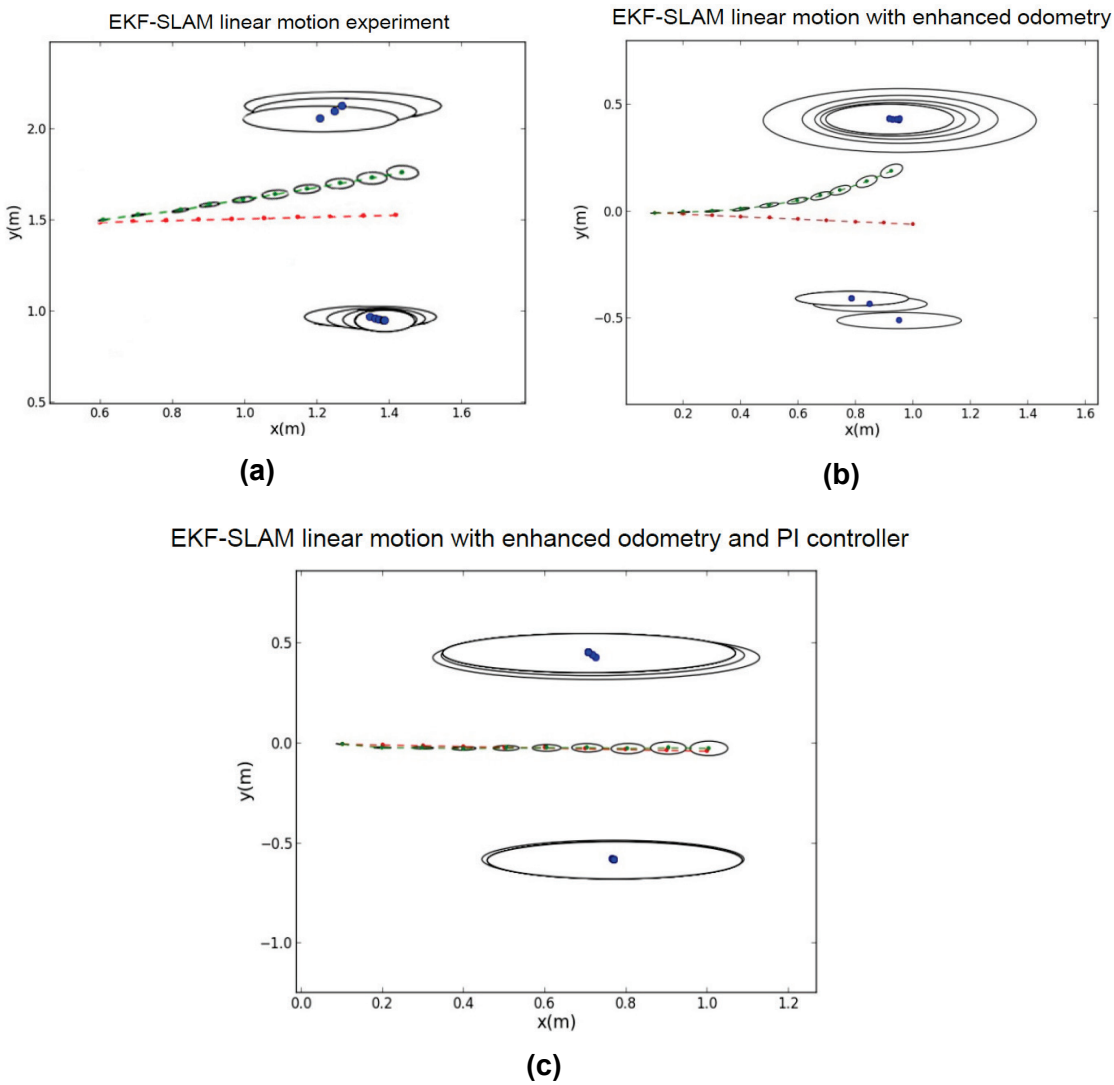


Figure 6.1. Comparison of results in linear motion experiment

Table 6.1. Experiment results on linear motion experiment of EKF-SLAM and AR-EKF-SLAM

Landmark number: 2	*Figure6.1(a) EKF-SLAM*	*Figure6.1(b) EKF-SLAM & Odometry*	*Figure6.1(c) EKF-SLAM & Odometry& Controller*
Average Error (m)	0.0652	0.0812	**0.0083**
Iterations to Complete	10	10	10
Runtime(s)	52.735	51.405	54.683

6.2. Result comparison and analysis of rectangular motion avoiding obstacle experiment

The second comparison is provided based on the experiment result under the condition of three landmarks, rectangular motion path with obstacle avoidance. Figure 6.2(a) to Figure 6.2(d) demonstrate the step by step improvement of original EKF-SLAM algorthm from intergrating Augmented Reality, numariacal results on each of the plot are collected in Table 6.2. As a comparsion, Figure 6.2 (a) is the result from executing original EKF-SLAM algorithm. The full experiment is completed in 16 EKF-SLAM iterations or 1m 46.297s, which is yet to be improved. By adding NAOmark recognition function, the control motion U is variant and related to the next landmark location, result depicted in Figure 6.2 (b), where the number of iterations required to finish the navigation task has been reduced from 16 to 11, along with a reduced runtime 1m 18.899s. Notice that the robot estimated position data is mainly based on the on-board robot odometry and is believed not reliable. That is to say, even though the robot estimated position appears to return at somewhere fairly closed to the reference location according to the plots, the actual robot position may be largely different. Accordingly, a more précised robot odometry can greatly improve the algorithm. In the next step, the robot odometry is replaced and updated using a new calculation method based on iPhone gyrometer data. The result of this experiment is shown in Figure 6.2 (c) characterizing and the randomness direction of robot trajectory and Table 6.2 presents the increase of measured robot position error according to the enhanced odometry. Technically, the increased position error should be more closed to the actual error than measuring from robot on-board odometry data, since the reliability of this update odometry has been verified through the experiment observation. Lastly, great robot

position improvement has been observed from Figure 6.2 (d) by integrating PI motion controller, with the reduction of error down to 0.0423m. As a result, the robot was able to return to the assigned location much closed than without using the enhanced odometry and PI motion controller.

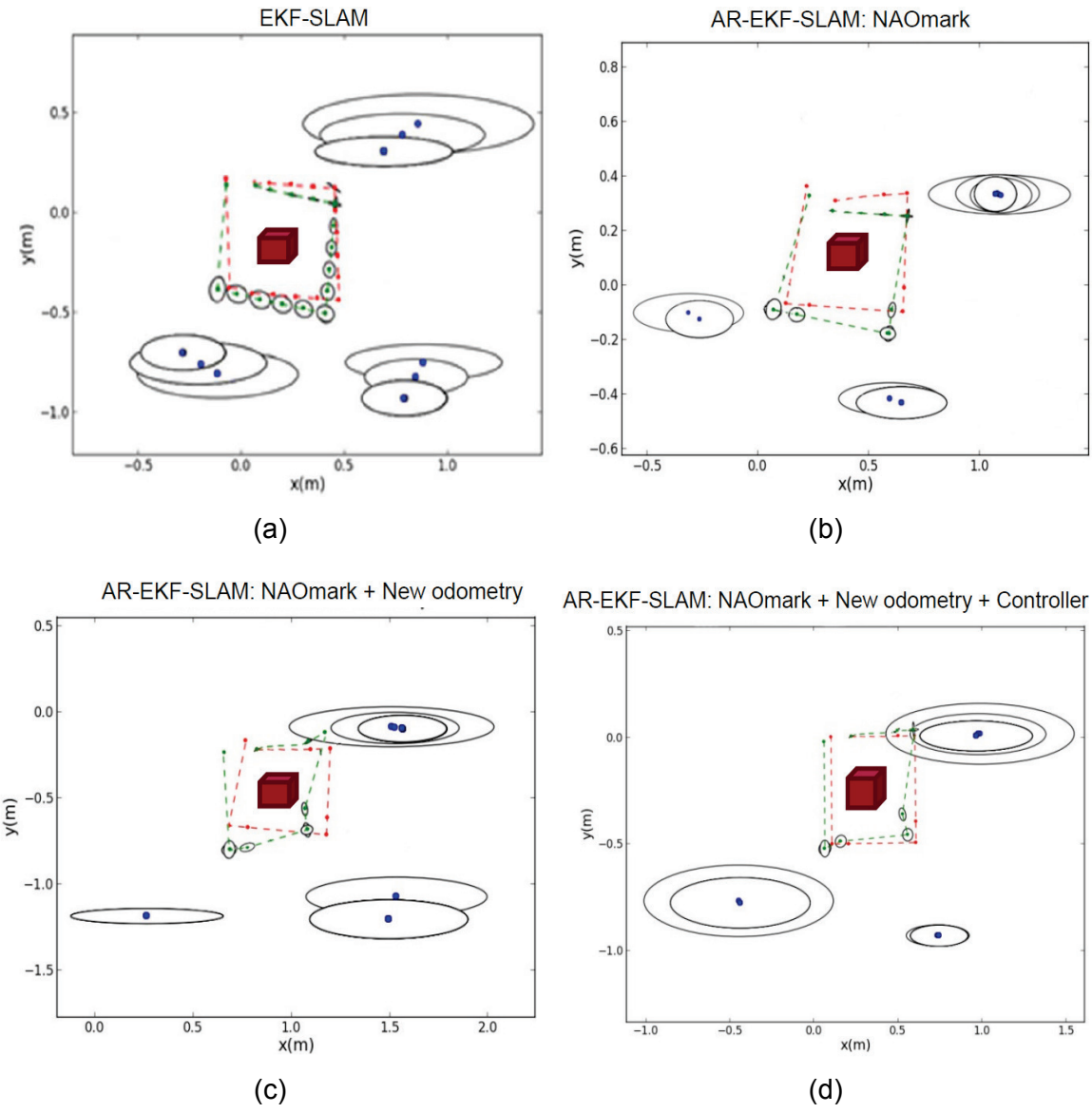


Figure 6.2. Comparison of results in Rectangular motion avoiding obstacle experiment

Table 6.2. Comparison of experiment results on EKF-SLAM and AR-EKF-SLAM

Landmark number: 3	*Figure6.2(a) EKF-SLAM*	*Figure6.2(b) AR-EKF-SLAM: NAOmark*	*Figure6.2(c) AR-EKF-SLAM: NAOmark + Odometry*	*Figure6.2(d) AR-EKF-SLAM: NAOmark + Odometry+ Controller*
Average Error (m)	0.07724	0.0808	**0.1542**	**0.0423**
Iterations to Complete	**16**	11	11	11
Runtime(s)	**1.46.297**	1.18.899	1.20.544	1.15.275

6.3. Summary

This chapter summaries the experimental results of EKF-SLAM and Augmented Reality integrated EKF-SLAM and then provides a comparison study in order to demonstrate the improvement. The comparison is made based on two experiment scenarios with their result plots and corresponding numerical results of average error, number of iterations and runtime. The first scenario, executing EKF-SLAM with two landmark and linear path motion, tested and verified the reliability of the enhanced odometry system based on iPhone gyroscope sensor data, as well as the performance of motion controller. The second comparison, under the scenario with full feature of Augmented Reality experiment, presents the improvement from each applications and the demonstration toward the final outcome of this thesis project, a full AR-EKF-SLAM.

Chapter 7.

Conclusions and Future Work

This thesis presented a real-time EKF-SLAM implementation on a humanoid robot NAO and an extended study of integrating Augmented Reality applications with EKF-SLAM. Improvements in terms of reducing EKF-SLAM computation expense and robot positioning are achieved.

In the first stage, EKF-SLAM algorithm was realized in Python code and studied in simulation and experimental implementation. The simulation result validated the effort of EKF-SLAM landmark observation in terms of reducing the uncertainty of robot motion. Two sets of full real-time EKF-SLAM implementation experiments were succeeded. The experiment one is the realization of simulation in two landmarks case, and the NAO robot in the experiment successfully accomplished EKF-SLAM in a realistic exploration task with three landmarks and fixed obstacle. The results for both experiment has verified the effectiveness of EKF-SLAM algorithm.

The next stage is considered as the essential contributions of this thesis. Two applications inspired by the fundamental concept of Augmented Reality are developed and integrated with EKF-SLAM. Application one utilizes pre-loaded information received from NAOmarks vision recognition function to improve the robot navigation and reduce EKF-SLAM computation cost. The second application was achieved in developing an enhanced robot odometry by applying external sensor data from iPhone gyrometer to EKF-SLAM's motion module. Additionally, the odometry data was used to calculate the robot position error and input into a closed-loop motion controller. As applications tested in experiments, the robot was able to complete the EKF-SLAM task in a realistic environment with reduced runtime and improved trajectory control.

7.1. Contributions

This thesis has presented a novel implementation of Augmented Reality technology onto humanoid robot NAO aiming to improve the performance of Extended Kalman Filter based Simultaneous Localization and Mapping algorithm. The work in this thesis has led to following contributions:

- A completed AR-EKF-SLAM code in Python programming language that is fully comparable to NAO humanoid robot
- Full experimental implementations of AR-EKF-SLAM algorithm on NAO robot was achieved in a realistic environment. The advantages of using Augmented Reality in EKF-SLAM is the reduction of computation expense and improvement of robot positioning, which have been claimed in the comparison chapter.
- A comprehensive overview of NAO humanoid robot in its specification, software framework and implementation methods

7.2. Recommendations for Future Work

Based on the achievements presented in this thesis, suggested future works are:

- Extend the use of NAO's camera by fusing vision based data with laser data to improve the EKF-SLAM observation function.
- Enrich the pre-loaded information from NAOmark recognition to enable the robot to do a multitude of navigation tasks.
- Conduct further experiments in more complex environments. i.e increasing the number of NAOmarks and obstacles to avoid.
- Improve the existed robot odometry by fusing data of gyroscope with accelerometers. i.e. equipping the robot with dedicated inertial units to obtain more precise location information.
- Implement a more reliable PI motion controller such as Fractional Order PI controller proposed in paper [11].

References

[1] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre and B. Maisonnier. The nao humanoid: A combination of performance and affordability. *CoRR Abs/0807.3223* 2008.

[2] G. Brooks, P. Krishnamurthy and F. Khorrami. Humanoid robot navigation and obstacle avoidance in unknown environments. Presented at Control Conference (ASCC), 2013 9th Asian. 2013, .

[3] J. Chestnutt, Y. Takaoka, K. Suga, K. Nishiwaki, J. Kuffner and S. Kagami. Biped navigation in rough environments using on-board sensing. Presented at Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference On. 2009, .

[4] P. Michel, J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner and T. Kanade. Online environment reconstruction for biped navigation. Presented at Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference On. 2006, .

[5] A. Yamashita, M. Kitaoka and T. Kaneko. Motion planning of biped robot equipped with stereo camera using grid map. *Interna-Tional Journal of Automation Technology 5(5),* pp. 639-647. 2011.

[6] R. Ozawa, Y. Takaoka, Y. Kida, K. Nishiwaki, J. Chestnutt, J. Kuffner, J. Kagami, H. Mizoguch and H. Inoue. Using visual odometry to create 3D maps for online footstep planning. Presented at Systems, Man and Cybernetics, 2005 IEEE International Conference On. 2005, .

[7] L. Yang and M. Worboys. Similarities and differences between outdoor and indoor space from the perspective of navigation. *Poster Presented at COSIT* 2011.

[8] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: Part I. *Robotics & Automation Magazine, IEEE 13(2),* pp. 99-110. 2006.

[9] K. Othman. "Implementation of EKF-SLAM on NAO humanoid robot" full version of thesis submitted to SFU.

[10] O. Mohareri and A. B. Rad. A visioin-based location positioning system via augmented reality: An application in humanoid robot navigation. *Internal Journal of Humanoid Robotics 10*pp. 1-26. (No.3) 2013.

[11] S. Wen, X. C. Y. Zhao, A. B. Rad, K. M. Othman and E. Zhang. "The study of fractional order controller with SLAM in the humanoid robot" Submitted to international journal of humanoid robotics.

[12] Y. Liu, X. Zhang, A. B. Rad, X. Ren and Y. Wong. Entropy based robust estimator and its application to line-based mapping. *Robotics and Autonomous Systems 58(5),* pp. 566-573. 2010.

[13] X. Zhang, A. B. Rad and Y. Wong. Sensor fusion of monocular cameras and laser rangefinders for line-based simultaneous localization and mapping (SLAM) tasks in autonomous mobile robots. *Sensors 12(1),* pp. 429-452. 2012.

[14] N. B. Ignell, N. Rasmusson and J. Matsson. An overview of legged and wheeled robotic locomotion.

[15] M. H. Hebert. *Intelligent Unmanned Ground Vehicles: Autonomous Navigation Research at Carnegie Mellon* 1997.

[16] T. Arlington and W. Hall. Unmanned aerial vehicles. 2010.

[17] C. von Alt. Autonomous underwater vehicles. Presented at Autonomous Underwater Lagrangian Platforms and Sensors Workshop. 2003, .

[18] R. C. Ooi. Balancing a two-wheeled autonomous robot. *University of Western Australia 3*2003.

[19] D. P. Anderson. nBot balancing robot. *2010-03-01) [2010-03-02]. Http: //Www.Smu.Edu/-Dpa-Www/Robo/Nbot/Index.Html* 2007.

[20] G. Lucas. A tutorial and elementary trajectory model for the differential steering system of robot wheel actuators. *The Rossum Project* 2001.

[21] S. Böttcher. Principles of robot locomotion. Presented at Proc. Human Robot Interaction Seminar SS. 2006, .

[22] R. A. Brooks, C. Breazeal, M. Marjanović, B. Scassellati and M. M. Williamson. "The cog project: Building a humanoid robot," in *Computation for Metaphors, Analogy, and Agents*Anonymous 1999, .

[23] (31 August 2012). *Four Legs*. Available: http://en.wikibooks.org /wiki/Robotics /Types_of_Robots/Walkers.

[24] M. Buehler. *Dynamic Locomotion with One, Four and Six-Legged Robots* 2005.

[25] D. R. Montello, "Navigation. in P. shah & A. miyake (eds.) The cambridge handbook of visuospatial thinking," in , Cambridge University Press., Ed. 2005, pp. 257-294.

[26] F. Lu. *Shape Registration using Optimization for Mobile Robot Navigation* 1995.

[27] P. Newman. On the structure and solution of the simultaneous localisation and map building problem. *Doctoral Diss., University of Sydney* 1999.

[28] M. Golfarelli, D. Maio and S. Rizzi. Correction of dead-reckoning errors in map building for mobile robots. *Robotics and Automation, IEEE Transactions On 17(1),* pp. 37-47. 2001.

[29] J. O. Wallgrun, "Spatial Representation and Reasoning for Mobile Robots," *Springer,Hierarchical Voronoi Graphs,* pp. 218p,Hardcover, 2010.

[30] J. J. Leonard, H. F. Durrant-Whyte and I. J. Cox. Dynamic map building for an autonomous mobile robot. *The International Journal of Robotics Research 11(4),* pp. 286-298. 1992.

[31] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions On 7(3),* pp. 376-382. 1991.

[32] R. Smith, M. Self and P. Cheeseman. "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles*Anonymous 1990, .

[33] S. Thrun. Robotic mapping: A survey. *Exploring Artificial Intelligence in the New Millennium* pp. 1-35. 2002.

[34] 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. *Robotics and Automation, IEEE Transactions On 17(3),* pp. 229-241. 2001.

[35] R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research 5(4),* pp. 56-68. 1986.

[36] P. Moutarlier and R. Chatila. An experimental system for incremental environment modelling by an autonomous mobile robot. Presented at Experimental Robotics I. 1990, .

[37] P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environment modeling. Presented at 5th International Symposium on Robotics Research. 1989, .

[38] J. Sola. Simulataneous localization and mapping with the extended kalman filter. 2013.

[39] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research 23(7-8),* pp. 693-716. 2004.

[40] J. Aulinas, Y. R. Petillot, J. Salvi and X. Lladó. The SLAM problem: A survey. Presented at CCIA. 2008, .

[41] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. Presented at AAAI/IAAI. 2002, .

[42] M. Montemerlo, S. Thrun and B. Siciliano. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics* 200727.

[43] W. Burgard, D. Fox, H. Jans, C. Matenar and S. Thrun. Sonar-based mapping with mobile robots using EM. Presented at MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-. 1999, .

[44] Z. Chen, J. Samarabandu and R. Rodrigo. Recent advances in simultaneous localization and map-building using computer vision. *Adv. Rob. 21(3-4),* pp. 233-265. 2007.

[45] P. Milgram and F. Kishino. A taxonomy of mixed reality visual displays. *IEICE Trans. Inf. Syst. 77(12),* pp. 1321-1329. 1994.

[46] R. Silva, J. Oliveira and G. Giraldi. *Introduction to Augmented Reality* 2003.

[47] A. Van Dam, A. S. Forsberg, D. H. Laidlaw, J. J. LaViola Jr and R. M. Simpson. Immersive VR for scientific visualization: A progress report. *Computer Graphics and Applications, IEEE 20(6),* pp. 26-52. 2000.

[48] W. R. Sherman and A. B. Craig. *Understanding Virtual Reality: Interface, Application, and Design* 2002.

[49] J. A. Vince. *Introduction to Virtual Reality* 2004.

[50] C. A. Pickover and S. K. Tewksbury. *Frontiers of Scientific Visualization* 1994.

[51] T. B. Sheridan. Musings on telepresence and virtual presence. *Presence: Teleoperators and Virtual Environments 1(1),* pp. 120-126. 1992.

[52] A. OLWAL. AN INTRODUCTION TO AUGMENTED REALITY.

[53] R. T. Azuma. A survey of augmented reality. *Presence 6(4),* pp. 355-385. 1997.

[54] R. Azuma. Tracking requirements for augmented reality. *Commun ACM 36(7),* pp. 50-51. 1993.

[55] R. L. Holloway. *Registration Errors in Augmented Reality Systems* 1995.

[56] O. Bimber, R. Raskar and M. Inami. *Spatial Augmented Reality* 2005.

[57] D. Van Krevelen and R. Poelman. A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality 9(2),* pp. 1. 2010.

[58] A. Takagi, S. Yamazaki, Y. Saito and N. Taniguchi. Development of a stereo video see-through HMD for AR systems. Presented at Augmented Reality, 2000.(ISAR 2000). Proceedings. IEEE and ACM International Symposium On. 2000, .

[59] J. Rekimoto. A magnifying glass approach to augmented reality systems. *Presence 6(4),* pp. 399-412. 1997.

[60] J. Rekimoto and K. Nagao. The world through the computer: Computer augmented interaction with real world environments. Presented at Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology. 1995, .

[61] D. Willers. Augmented reality at airbus. Presented at International Symposium on Mixed & Augmented Reality. 2006, .

[62] W. Friedrich, D. Jahn and L. Schmidt. ARVIKA-augmented reality for development, production and service. Presented at ISMAR. 2002, .

[63] I. Kasai, Y. Tanijiri, T. Endo and H. Ueda. A forgettable near eye display. Presented at Wearable Computers, the Fourth International Symposium On. 2000, .

[64] V. Hayward, O. R. Astley, M. Cruz-Hernandez, D. Grant and G. Robles-De-La-Torre. Haptic interfaces and devices. *Sens Rev 24(1),* pp. 16-29. 2004.

[65] H. Fuchs, M. A. Livingston, R. Raskar, K. Keller, J. R. Crawford, P. Rademacher, S. H. Drake and A. A. Meyer. "Augmented reality visualization for laparoscopic surgery," in *Medical Image Computing and Computer-Assisted Interventation—MICCAI'98*Anonymous 1998, .

[66] T. Ohshima, K. Satoh, H. Yamamoto and H. Tamura. AR< sup> 2 hockey: A case study of collaborative augmented reality. Presented at Virtual Reality Annual International Symposium, 1998. Proceedings., IEEE 1998. 1998, .

[67] J. Chestnutt, K. Nishiwaki, J. Kuffner and S. Kagami. Interactive control of humanoid navigation. Presented at Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference On. 2009, .

[68] M. Shoji, K. Miura and A. Konno. U-tsu-shi-O-mi: The virtual humanoid you can reach. Presented at ACM SIGGRAPH 2006 Emerging Technologies. 2006, .

[69] (Apr. 25, 2013). *aldebaran robotics website*. Available: http://www.aldebaranrobotics. com/en/Home/welcome.html?language=en-GB.

[70] (May 15, 2013). *NAO Software 1.14 Documentation*. Available: http://www.aldebaranrobotics. com/documentation/index.html.

[71] T. Deutsch, C. Muchitsch, H. Zeilinger, M. Bader, M. Vincze and R. Lang. Cognitive decision unit applied to autonomous biped robot nao. Presented at Industrial Informatics (INDIN), 2011 9th IEEE International Conference On. 2011, .

[72] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre and B. Maisonnier. The nao humanoid: A combination of performance and affordability. *CoRR Abs/0807.3223* 2008.

[73] T. González Sánchez. Artificial vision in the NAO humanoid robot. 2009.

[74] S. Riisgaard and M. R. Blas. SLAM for dummies. *A Tutorial Approach to Simultaneous Localization and Mapping 22*pp. 1-127. 2003.

[75] A. Gil, O. Reinoso, O. M. Mozos, C. Stachnissi and W. Burgard. Improving data association in vision-based SLAM. Presented at Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference On. 2006, .

[76] G. Fang. Data association in bearing-only SLAM using a cost function-based approach. 2007.

[77] A. Manz, R. Liscano and D. Green. *A Comparison of Realtime Obstacle Avoidance Methods for Mobile Robots* 1993.

[78] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. Presented at Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference On. 1991, .

[79] L. Wang, J. Shi, G. Song and I. Shen. "Object detection combining recognition and segmentation," in *Computer Vision–ACCV 2007*Anonymous 2007, .

[80] (2013-04-29). *ALLandMarkDetection Overview*. Available: http://www.aldebaran-robotics.com/documentation/naoqi/vision/allandmarkdetection.html#allandmarkdetection.