SENSOR BASED LOCALIZATION OF A MOBILE ROBOT

Lee Gim Hee
B.Eng (Hons), NUS

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2007

ACKNOWLEDGMENTS

The author wishes to express his heart-felt gratitude to his supervisor, Associate Professor Marcelo H. Ang Jr for his guidance through the years. He is grateful to Professor Ang for providing him with a lot of opportunities to extend his knowledge and to develop his skills.

He would like to show his appreciation to those dedicated technicians from the Control and Mechatronics Laboratory of the National University of Singapore who have assisted him in the course of his project. He is glad to work with a great team of fellow research students in his laboratory. They are Mana Saedan, Koh Niak Wu, Tirthankar Bandyopadhyay and many other individuals who have contributed valuable ideas and advice to make this thesis possible.

The author is indebted to his caring parents who are always doing their best in giving him the finest environment to do his Masters of Engineering work. He would also like to express thanks to his sister and brother for being supportive. Finally, the author would like to thank his wife, Grace Tang who is always by his side. Her unconditional love has always been a source of strength, courage, inspiration and happiness to the author.

SUMMARY

Localization is the process of determining the pose of a mobile robot with respect to a given map of the environment (known environment). The localization problem can be made more difficult in cases where the map of the environment is not given (unknown environment) to the robot. This is the *simultaneous localization and mapping* (SLAM) problem where the robot has to simultaneously build a map of its environment and localizes itself with respect to this map.

Some of the sensors that are commonly used for mobile robot localization are evaluated in this thesis. The odometer and laser range finder are found to be the most suitable sensors for implementing the localization algorithms. A probabilistic algorithm - *The Particle Filter* has been chosen over other algorithms for localization of a mobile robot in a known environment due to its robustness, effectiveness and ease of implementation.

The most significant contribution of this thesis is a novel solution for the SLAM problem. This novel SLAM algorithm uses a laser scan matching algorithm to align consecutive laser scans, loop closure detection algorithm to detect loop closure opportunity, and loop closure algorithm to close any detected loops in the map.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

NOMENCLATURE

$\mathbf{x_t}$	Robot pose $[x\ y\ \theta]^T$ where x, y and θ are the cartesian coordinates of the robot location and orientation with respect to a global fixed reference frame
$\mathbf{z_t}$	Sensor measurement data
$\mathbf{z_t^*}$	Predicted sensor measurement data
z_t^k	k^{th} measurement reading provided by a sensor
$\mathbf{u_t}$	Control data
$bel(\mathbf{x_t})$	Belief distribution of the of a robot, also denoted by $p(\mathbf{x_t} \mid \mathbf{z_{1:t}},\ \mathbf{u_{1:t}})$
$\overline{bel}(\mathbf{x_t})$	Predicted belief distribution of the of a robot
$p(\mathbf{x_t} \mid \mathbf{x_{t-1}},\ \mathbf{u_t})$	State transition probability
$p(\mathbf{z_t} \mid \mathbf{x_t})$	Measurement probability
μ	Mean of a multi-dimensional variable that follows Guassian distribution
$\mathbf{\Sigma}$	Covariance of a multi-dimensional variable that follows Guassian distribution
σ	Standard deviation of an one-dimensional variable that follows Guassian distribution
η	Probability normalizer

n_i	Number of revolutions taken by the left or right robot wheel, i = left or right
D	Diameter of the robot wheel
d_i	Linear distance traveled by the left or right robot wheel, i = left or right
L_{axle}	Distance between the floor contact points of the left and right wheels
ϑ	Map of robot's environment
ξ_t	Finite sample set of particles
$\chi_t^{[m]}$	The m^{th} particle $[\mathbf{x_t^{[m]}}\ w_t^{[m]}]^T$ where $\mathbf{x_t^{[m]}}$ denotes its hypothetical state and $w_t^{[m]}$ denotes its importance factor
δ_{trans}	Translation of the robot
δ_{rot}	Rotation of the robot
$\mathbf{x_t^{max}}$	Maximum weight particle
$\mathbf{x_t^{estimate}}$	Pose estimate from particle filter algorithm
β	Radius of a circular window for determining the pose estimate from particle filter
$\tilde{w}_t^{[i]}$	The weight of the particle $\mathbf{x_t^{[i]}}$ normalized over the total weight of all the particles that are enclosed within the circular window of radius β and center $\mathbf{x_t^{max}}$
$\mathbf{P_t}$	Error covariance of the state estimate for localization with extended Kalman filter
$\mathbf{f}(.)$	Motion model for the extended Kalman filter
$\mathbf{F}$	Jacobian of $\mathbf{f}(.)$ evaluated at $\mathbf{x_{t-1}}$
$\mathbf{Q_t}$	Error covariance of the motion model for the extended Kalman filter

$\mathbf{K_t}$	Kalman gain
$\mathbf{h}(.)$	Measurement model for the extended Kalman filter
$\mathbf{H_t}$	Jacobian of $\mathbf{h}(.)$ evaluated at $\mathbf{x_{t-1}}$
$\mathbf{R_t}$	Error covariance of the measurement model for the extended Kalman filter
$p(\mathbf{x_t}, \vartheta \mid \mathbf{z_{1:t}}, \mathbf{u_{1:t}})$	Joint probability distribution of the robot state $\mathbf{x_t}$ and the map ϑ of the environment given all the measurement data $\mathbf{z_{1:t}}$ and control data $\mathbf{u_{1:t}}$
$\mathbf{y_t}$	Combined state vector for SLAM with extended Kalman filter
$\overline{\mathbf{S}}_\mathbf{t}$	Error covariance of combined state vector estimate for SLAM with extended Kalman filter
$p(\vartheta_i \mid \mathbf{z_{1:t}}, \mathbf{x_{1:t}})$	Occupancy value of the i^{th} grid cell in an occupancy grid map
$l_{i,t}$	Log odds of the i^{th} grid cell in an occupancy grid map
ε_k	Binary operator for determining if the k^{th} measurement reading should be included for scan matching
γ	Distance threshold for adding new nodes in the topological map for loop closure detection
$\mathbf{x_E}$	True pose of robot for loop closure
$\mathbf{x_S}$	Pose from the trajectory that was previously taken by the robot where the loop closure opportunity is detected
$\mathbf{x}_{\mathbf{S:t}}^{\mathbf{corrected}}$	Corrected trajectory after loop closure
$\mathbf{x}_{\mathbf{S:t}}^{\mathbf{forward}}$	Forward trajectory for loop closure
$\mathbf{x}_{\mathbf{S:t}}^{\mathbf{backward}}$	Backward trajectory for loop closure
α_k	Weighing factor for the forward and backward trajectories

CHAPTER 1

INTRODUCTION

1.1 Background

The problem of autonomy of a mobile robot is simply summarised by Leonard and Whyte as the necessary solutions to the following three questions: "Where am I?", "Where am I going?", and "How should I get there?" [1]. The first problem, which is the focus of this thesis, defines the localization problem. Localization can be seen as the process of determining the pose (position and orientation) of the mobile robot with respect to a global frame of reference. A mobile robot is able to determine its destination and plan a path that will enable it to navigate there safely only if it is capable of finding its own location at every instance of time. In other words, the "Where am I?" problem has to be answered before the "Where am I going?" and "How should I get there?" problems could be solved.

The earliest solution to the localization problem is to do relative pose measurement [2] using sensors such as odometer, or *inertia measurement unit* (IMU). Relative pose measurement, otherwise known as dead reckoning, is the process of tracking the current pose of the robot based on the integration of the path that has been previously traveled by the robot. However, the pitfall of relative pose measurement using sensors

is that the systematic and random errors on the sensors are also integrated into the pose measurement. The accumulated errors will eventually grow unbounded and thus the pose measurement becomes grossly erroneous. Examples of systematic errors in odometry are unequal robot wheel diameters and finite encoder resolution. Random errors in odometry include wheel slippage and traveling across uneven floors. Some researchers tried to improve the odometry readings by incorporating error models into the odometry readings. One of the approach is the UMBmark test [2, 3, 4] proposed by Borenstein and Feng. The UMBmark test is based on a set of well-defined experimental procedures that aimed to quantify the measurement of systematic odometry errors and, to a limited degree, random odometry errors.

Another early approach to the localization problem is to do relative pose measurement [2] using active beacons. Localization using active beacons has been traditionally used in the *global positioning system* (GPS) for the localization of ships and airplanes. There are two types of active beacon systems: trilateration and triangulation [5, 6]. Trilateration is the determination of the robot position based on distance measurements to known beacon sources. In trilateration localization systems there are usually three or more transmitters mounted at known locations in the environment and one receiver fixed onto the robot. Note that the orientation of the mobile robot is not found in trilateration. An example of the active beacons localization system that make use of trilateration is the Cricket Motes [7]. Triangulation is the determination of the robot pose based on the measurements of the angle from the beacon to the robot heading. The distance to at least one of beacons and its location must also be known. Similar to the trilateration method, three or more beacon readings must be

obtained to do triangulation. Note that it is also possible to do triangulation with two beacons if the angles from these beacon to the robot heading, the distances from these beacons to the robot, and the locations of these beacons are known (see Section 3.2.6 for more details). An example of the triangulation with two beacons is the NorthStar localization kit [8, 9]. Unlike dead reckoning, the errors in active beacons localization systems will not grow unbounded. However, the accuracy is highly dependent on the size of its random errors and precise placement of the beacons in the environment.

It has become apparent from both the relative and absolute position measurement examples that no deterministic approach is capable of providing the accurate pose estimate of the mobile robot. This is largely attributed to noisy sensor readings that carry only partial information of the measured variables. Hence, many researchers turned to probabilistic approaches [1, 10, 11, 12, 13, 14, 15, 16]. The key idea of probabilistic localization is to assign probability values to each hypothesis of the robot pose from probability density functions conditioned upon the sensory data. In other words, the pose of a mobile is represented by probability distributions over a whole space of guesses instead of relying on a single "best guess".

In this thesis, probabilistic algorithms shall be used to solve the localization problem. First, some basic concepts of probabilistic mobile robot localization are discussed. The Bayes filter, which is the most general form of probabilistic algorithm to recursively estimate the pose of a mobile robot, will also be reviewed. Next, some of the sensors that are commonly used for mobile robot localization will be evaluated. The sensors which are evaluated include *inertia measurement unit* (IMU), compass,

global positioning system (GPS), odometer, Cricket Motes, NorthStar localization kit and laser range finder. The best sensor(s) for implementation of the localization algorithms on the ER2 mobile robot will be selected. One of the probabilistic algorithm to solve the localization problem - the particle filter algorithm [14, 15, 16, 17] will be examined. The algorithm will be simulated in virtual environments and implemented on the ER2 mobile robot.

The localization problems described so far has been in the context where a mobile robot has to locate its pose with reference to a known model of the environment. The problem could be made more difficult if the model of the environment is not available to the robot. This is the *simultaneous localization and mapping* (SLAM) problem [1, 10, 15, 18, 19, 20, 21, 22]. The highlight of this thesis is a novel algorithm proposed by the author to solve the SLAM problem. The algorithm is successfully implemented on the ER2 mobile robot.

1.2 Objectives

The objective of this Masters of Engineering work is to investigate and implement localization algorithms on the ER2 mobile robot shown in Figure 1.1. Note that the robot is named ER2 because it is a modification of the ER1 robot from Evolution Robotics, Inc[1]. The implemented algorithms give the robot the capability to do local localization, global localization, solve the kidnapped (see Chapter 4 for details) and SLAM (see Chapter 5 for details) problems. All localization algorithms are carried

[1]Company webpage: http://www.evolution.com/

out in indoor environments.

Figure 1.1: The ER2 mobile robot with Joystick control.

1.3 Scope of Work

The scope of work includes the following:

1. Evaluate and select the best sensor(s) for the robot to perform indoor localization.

2. To investigate and implement localization algorithms for the robot to compute its pose with respect to a given map using its sensory data.

3. To provide a complete solution for a mobile robot to build a grid-based representation of its surrounding using its sensory data and localize itself with respect to this map.

4. All algorithms are to be implemented on the ER2 mobile robot and tested in indoor environments.

1.4 Contributions of This Thesis

This thesis gives a concise analysis of the characteristics, advantages and disadvantages of some sensors that are commonly used for the localization of a mobile robot. The sensors which are evaluated include *inertia measurement unit* (IMU), compass, *global positioning system* (GPS), odometer, Cricket Motes, NorthStar localization kit and laser range finder.

Different variations of the particle filter are simulated and implemented on the ER2 mobile robot platform. The different variations of the particle filter are meant for solving the local localization, global localization and kidnapped problem respectively.

The most significant contribution of this research is in solving the SLAM problem. A novel algorithm is proposed and implemented on the ER2 mobile robot to do SLAM. This algorithm uses a laser scan matching algorithm to align consecutive laser scans, a loop closure detection algorithm to detect loop closure opportunity and

loop closure algorithm to close any detected loops in the map.

1.5 Thesis Outline

The outline of this thesis is as follows:

Chapter 2 This chapter gives the definitions of some terms that are commonly used in the mobile robot localization context. The Bayes filter, which is the most general form of probabilistic algorithm to recursively estimate the pose of a mobile robot, will be reviewed.

Chapter 3 This chapter gives a detailed analysis of the characteristics, advantages and disadvantages of some sensors that are commonly used for the localization of a mobile robot. Evaluations are carried out on IMU, compass, GPS, odometer, Cricket Motes, NorthStar localization kit and laser range finder. The best sensor(s) for the robot to perform localization in an indoor environment will be selected.

Chapter 4 This chapter defines the local localization, global localization and kidnapped problem of a mobile robot in a known environment. The particle filter is discussed in detail. An overview of the particle filter is first presented. Next, the motion and measurement models of the filter are discussed. Different variations of the particle filter are suggested to solve the three localization problems in known indoor environments. Results from simulations and implementations of the algorithms will

be shown in this chapter.

Chapter 5 This chapter describes a novel algorithm to solve the SLAM problem. This algorithm uses a laser scan matching algorithm to align consecutive laser scans, a loop closure detection algorithm to detect loop closure opportunity and loop closure algorithm to close any detected loops in the map. Results from the implementations of the algorithm on the ER2 mobile robot will be shown here. A brief description of the occupancy grid mapping algorithm will also be given.

Chapter 6 This chapter gives the conclusion of the work. Some possible further works are also discussed in this chapter.

CHAPTER 2

BASIC CONCEPTS

2.1 Basic Concepts in Mobile Robot Localization

This section introduces the definitions of some commonly used terms in the literature of mobile robot localization. The pose of the mobile robot, which is also known as the state of the robot, will be defined first. Next, sensor measurements and control actions which are both fundamental means for the robot to interact with the environment are defined. These definitions are adapted from [10, 23].

2.1.1 State

A mobile robot can be represented as rigid bodies in an Enclidean workspace, W$\in \Re^N$ where N equals to 2. Hence, the pose of a mobile robot at time t denoted by $\mathbf{x_t}$ can be fully defined by three variables. The three variables are the position coordinates (x, y) and heading direction θ of the robot defined with respect to a fixed global coordinate frame in the Enclidean workspace. The pose of a mobile robot will also be referred to as *state* in this thesis. Equation 2.1 shows the mathematical denotation for the state of a mobile robot at time t.

$$\mathbf{x_t} = [x \quad y \quad \theta]^T \tag{2.1}$$

Figure 2.1 shows an illustration of the robot pose with respect to a global fixed frame.

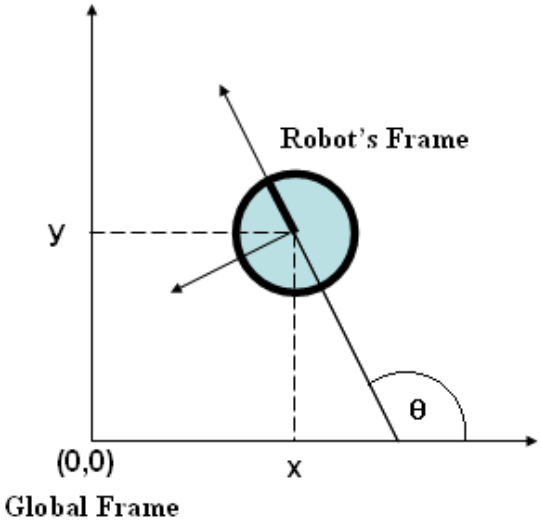


Figure 2.1: The pose of a mobile robot with respect to a global fixed frame.

2.1.2 Sensor Measurements

A mobile robot gains momentary perception of its surrounding environment via interpretations made from its sensor measurement data. For example, laser range finders give the robot range information of objects in its environment. Some sensory data may be available to the robot with some delay. In this research, it is assumed that a new set of measurements is always available to the controller of the robot at every

instance of sampling. This is usually a valid assumption because the measurement data of most sensors are updated at a much faster rate than the rate at which the robot controller acquires the data. The measurement data set provided by a sensor at time t will be represented mathematically by Equation 2.2.

$$\mathbf{z_t} = [z_t^1 \quad z_t^2 \quad ... \quad z_t^k]^T \tag{2.2}$$

where z_t^k is the k^{th} measurement reading provided by the sensor. For example, $k = [1,\ 360]$ for a laser range finder that provides 360 measurement data per scan.

2.1.3 Control Actions

A mobile robot changes its state by executing control actions that exert forces on its environment. For example, a mobile robot can change its state by exerting forces to turn its wheel for motion. In this thesis, information about the change of state of the robot due to the control actions shall be given by the control data.

A typical control data is the velocity of the mobile robot because this information can be used to compute the change of state of the robot. For example, a velocity of 10 cm/s, indicates a movement of 100cm away from the previous pose, in the direction of the velocity after 10 seconds.

Another possible control data is the odometry reading. Odometry readings are provided by odometer which return the current pose of the robot with respect to an initial frame of reference by counting the number of revolutions in the robot's wheels

(see Section 3.2.4 for more details). As such, odometry readings convey information about the change of state.

Control data will be denoted by $\mathbf{u_t}$. The variable $\mathbf{u_t}$ corresponds to the change in the pose of the mobile robot in the time interval $(t-1,\ t]$ due to a control action. It is assumed that there will always be exactly one control data per time step t in this research. Note that the control data is also available even in the event where the robot does not move. This control action is taken as an instruction for the robot to "do nothing".

In this research, the odometry readings shall be used as the control data. The readings from odometers are generally more accurate than velocity because in addition to the drift and slippage error, velocity suffers inaccuracies in the computation of the actual change of state using some mathematical model.

2.2 Recursive State Estimation

It was mentioned in Section 1.1 that deterministic approaches are not capable of giving accurate estimations of the robot pose. This is due to noisy sensory data that carry only partial information of the measured variables. Hence, it is desirable to rely on probabilistic approaches that estimate the robot state recursively over time. Recursive state estimation is a probabilistic approach for estimating the unknown probability density function of a state variable recursively over time using a mathematical process model and incoming sensory data. Some terms for recursive state

estimation will be defined in Sections 2.2.1 and 2.2.2. Finally, the Bayes filter which is the most general algorithm for doing recursive state estimation will be discussed in Section 2.2.3. These definitions are adapted from [10].

2.2.1 Belief Distributions

A belief distribution refers to the estimated probability density function of the state variable during the state estimation process. A belief distribution assigns a probability or density value to each possible hypothesis with regards to the true state of the robot. The hypothesis of the true state is otherwise known as the belief. A belief reflects the robot's internal knowledge about the state of itself. The belief distributions are posterior probabilities over the robot state $\mathbf{x_t}$ at time t conditioned on all past measurements $\mathbf{z_{1:t}}$ and all the past control data $\mathbf{u_{1:t}}$. In this thesis, the belief over the state variable $\mathbf{x_t}$ shall be denoted by $bel(\mathbf{x_t})$, which is an abbreviation for the posterior

$$bel(\mathbf{x_t}) = p(\mathbf{x_t} \mid \mathbf{z_{1:t}}, \mathbf{u_{1:t}}) \tag{2.3}$$

2.2.2 State Transition and Measurement Probabilities

The state transition probability specifies how the state of the robot evolves over time as a function of the current control data $\mathbf{u_t}$. The probability is denoted by $p(\mathbf{x_t} \mid \mathbf{x_{t-1}}, \mathbf{u_t})$. It is important to note that the current robot state $\mathbf{x_t}$ is independent of all the past control data $\mathbf{u_{1:t-1}}$ and sensor measurements $\mathbf{z_{1:t-1}}$. This is because the robot state $\mathbf{x}$ is assumed to be a complete state and hence $\mathbf{x_{t-1}}$ is

a sufficient statistic of all the previous control data and sensor measurements. The state transition probability illustrates the *Markov assumption*, which postulates that the past and future data are independent if the current robot state $\mathbf{x_t}$ is known.

The measurement probability specifies the probabilistic law according to which the measurements $\mathbf{z_t}$ are generated from the environment and current robot state $\mathbf{x_t}$. The probability is denoted by $p(\mathbf{z_t} \mid \mathbf{x_t})$. It is also important to note that the measurement probability is independent of all the past robot states $\mathbf{x_{1:t-1}}$, sensor measurements $\mathbf{z_{1:t-1}}$ and all the control data $\mathbf{u_{1:t}}$. This is again due to the assumption that the robot state $\mathbf{x_t}$ is complete. Hence, following the Markov assumption, the robot state $\mathbf{x_t}$ is sufficient to predict the current sensor measurement $\mathbf{z_t}$.

Both the state transition and measurement probabilities are taken to be independent of the time index t in this thesis. This means that the probability density functions of both $p(\mathbf{x_t} \mid \mathbf{x_{t-1}}, \mathbf{u_t})$ and $p(\mathbf{z_t} \mid \mathbf{x_t})$ do not change over time. This assumption implies that both the probability density functions of $p(\mathbf{x_t} \mid \mathbf{x_{t-1}}, \mathbf{u_t})$ and $p(\mathbf{z_t} \mid \mathbf{x_t})$ can be pre-determined and consistently used throughout the recursive state estimation process (see Sections 4.3.1 and 4.3.2 for more details).

An example of the probability density function for $p(\mathbf{x_t} \mid \mathbf{x_{t-1}}, \mathbf{u_t})$ and $p(\mathbf{z_t} \mid \mathbf{x_t})$ is the multivariate normal distribution given by the Gaussian function in Equation 2.4 with mean μ and covariance $\mathbf{\Sigma}$.

$$p(a) = det(2\pi\Sigma)^{-\frac{1}{2}} exp\{-\frac{1}{2}(a-\mu)^T\Sigma^{-1}(a-\mu)\} \tag{2.4}$$

In the case of where the random variable is one-dimensional, the normal distribution becomes

$$p(a) = \frac{1}{\sigma\sqrt{2\pi}} exp\{-\frac{(a-\mu)^2}{2\sigma^2}\} \tag{2.5}$$

where σ denotes the standard deviation. Figure 2.2 shows a plot of the one-dimensional normal distribution with mean and standard deviation equal to 0 and $\sqrt{3}$ respectively.

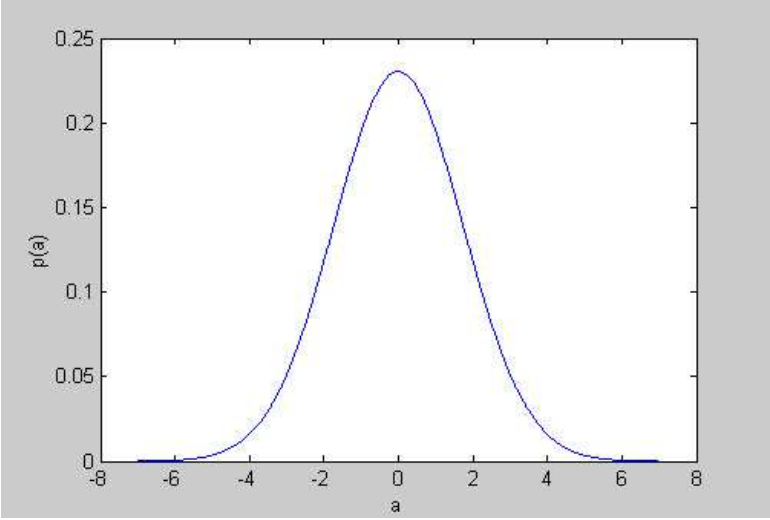


Figure 2.2: Normal distribution with $\mu = 0$ and $\sigma = \sqrt{3}$.

2.2.3 The Bayes Filter

The Bayes filter is the most general algorithm for doing recursive state estimation. It calculates the posterior belief distribution $bel(\mathbf{x_t})$ from the most recent control data $\mathbf{u_t}$ and sensor measurement $\mathbf{z_t}$. Table 2.1 shows the pseudo code for a single iteration

of the Bayes filter.

1: **Bayes_filter(**$bel(\mathbf{x_{t-1}})$, $\mathbf{u_t}$, $\mathbf{z_t}$):
2: **for** all $\mathbf{x_t}$ **do**
3: $\quad \overline{bel}(\mathbf{x_t}) = \int p(\mathbf{x_t} \mid \mathbf{u_t}, \mathbf{x_{t-1}})\ bel(\mathbf{x_{t-1}})\ d\mathbf{x_{t-1}}$
4: $\quad bel(\mathbf{x_t}) = \eta\ p(\mathbf{z_t} \mid \mathbf{x_t})\ \overline{bel}(\mathbf{x_t})$
5: **end for**
6: **return** $bel(\mathbf{x_t})$

Table 2.1: The Bayes filter algorithm.

There are two essential steps in the Bayes filter: the *prediction* and *update* step. Line 3 is the prediction step. In this step, the predicted belief distribution $\overline{bel}(\mathbf{x_t})$ is computed from the integral of the product of two distributions: the prior belief distribution $bel(\mathbf{x_{t-1}})$ and the state transition distribution $p(\mathbf{x_t} \mid \mathbf{x_{t-1}}, \mathbf{u_t})$.

Line 4 is the measurement update step. In this step, the current belief distribution $bel(\mathbf{x_t})$ is computed from the product of the measurement distribution $p(\mathbf{z_t} \mid \mathbf{x_t})$, predicted belief distribution $\overline{bel}(\mathbf{x_t})$ and a normalizing constant η. Note that the product of $p(\mathbf{z_t} \mid \mathbf{x_t})$ and $\overline{bel}(\mathbf{x_t})$ may not sum up to 1 and this violates the axiom of probability which states that all probability density distributions must integrate to 1 [24]. Hence, the normalizing constant η is needed to make sure that the posterior belief distribution $bel(\mathbf{x_t})$ sum up to 1.

An initial belief $bel(\mathbf{x_0})$ is required to do recursive state estimation using the Bayes filter. The initial belief should be initialized to a point mass distribution if $\mathbf{x_0}$

is known with certainty. Alternatively, the initial belief should be initialized using an uniform distribution over the domain of $\mathbf{x_0}$ if the initial value of $\mathbf{x_0}$ is completely unknown. Non-uniform distributions could also be used if partial knowledge of $\mathbf{x_0}$ is available.

It is important to note that it is not practical to implement the Bayes filter for the localization of a mobile robot despite the fact that it is the most general algorithm for recursive state estimation. This is because the state space $\mathbf{x_t}$ is continuous and therefore it is impossible to represent the beliefs in Lines 3 and 4 with a digital computer. Nevertheless, there exist a number of techniques and algorithms to overcome this problem. These techniques and algorithms rely on assumptions to approximate the belief distributions. An example is the Kalman filter which assumed that the belief distributions are Gaussian and hence Lines 3 and 4 can be calculated in closed form. Another example is the particle filter which seeks to represent the belief distribution with a finite number of samples known as the particles and therefore Lines 3 and 4 can be computed with a digital computer. The particle filter will be used to solve the localization problems in this thesis. See Chapters 4 and 5 for more details.

CHAPTER 3

SENSOR EVALUATION

3.1 Introduction

Sensors play an important role in the localization of mobile robots. This is because they provide information regarding the internal state of the robot and its environment which are essential for deducing the robot pose.

Many sensors are available in the market for mobile robot localization. For example, IMU, compass, GPS, odometer, Cricket Motes, NorthStar localization kit and laser range finder. However, the characteristics and operating principles of these sensors may limit their applications to only certain areas. For example, the radio frequency (RF) transmitted from satellites for GPS pose measurement gets attenuated by buildings, tree canopies or even clouds [25, 26]. Hence restricting its use in buildings, jungles or cloudy days. Other indoor active beacons systems such as Cricket Motes and NorthStar localization kit require modification to the environment, thus limiting its use to mobile robot localizations in known environments.

In this chapter, some of the sensors which are commonly used for mobile robot localization will be evaluated. These sensors include IMU, compass, GPS, odometer,

Cricket Motes, NorthStar localization kit and laser range finder. Note that ultrasonic range finder which is also a commonly used sensor in mobile robots for obstacles avoidance will not be included in the evaluation. This is because ultrasonic range finder are generally not accurate enough for mobile robot localization. The characteristics, advantages and disadvantages of the selected sensors shall be discussed. Finally, the best sensor(s) for robot localization in both known and unknown environments will be selected. The sensor(s)will be used for implementation of the localization algorithms on the ER2 mobile robot.

3.2 Sensor Review

In this section, the characteristics, advantages and disadvantages for some of the most commonly used sensors in mobile robot localization will be reviewed. These sensors include IMU, electronic compass, GPS, odometer, Cricket Motes, NorthStar Localization Kit and laser range finder.

3.2.1 Inertia Measurement Unit

The inertia measurement unit (IMU) is a single unit system that consists of both the accelerometer and gyroscope to detect accelerations along the x, y and z axis, and the rate of change in attitude (i.e. roll, pitch and yaw rates) respectively. The total change from the initial positions along the x, y and z axis are subsequently found by double integration of the acceleration along the respective axis. The total change from the initial roll, pitch and yaw angles are found by single integration of

the angular rate around the respective axis.

Figure 3.1 shows the RGA300CA IMU manufactured by Crossbow Technology, Inc[2]. This IMU consists of a high performance MEMS gyroscope and tri-axial accelerometer [27]. The sensor is designed to measure rotation rates around yaw axis and linear acceleration along the x, y and z axis. The MEMS angular rate sensor is mounted with a z-sensitive axis vibrating ceramic plates that utilize the Coriolis force to output angular rate independently of acceleration. The three MEMS accelerometers are surface micro-machined silicon devices that use differential capacitance to sense acceleration.

Figure 3.1: RGA300CA IMU from Crossbow Technology, Inc.

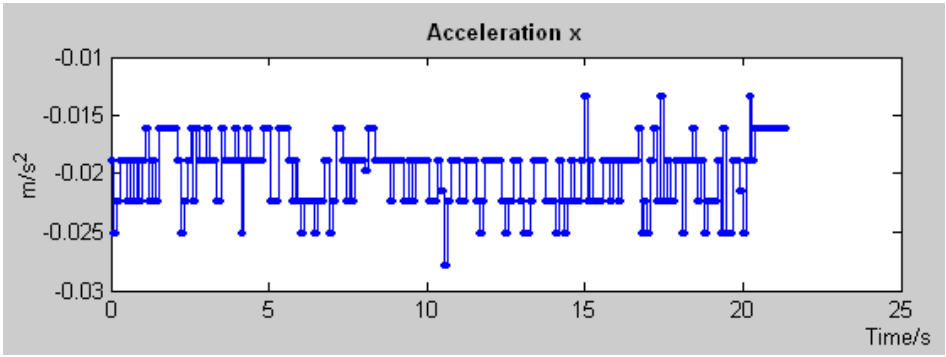
[2]Company webpage: http://www.xbow.com/

There are three main advantages of IMU. First, relative pose information are easily obtained from double and single integrations. Second, the sensor does not require any modifications of the environment. Third, IMU is suitable for both indoor and outdoor localization purposes. Unfortunately, IMU typically suffers from drifts where the signals change over time even if there are no changes in the acceleration. Furthermore, errors are accumulated during the integration process thus leading to an ever increasing error in the pose measurements. The rate of drift and accumulation of errors differs for different IMUs. Generally, IMUs with lower rate of drift and accumulation of errors tend to cost more.

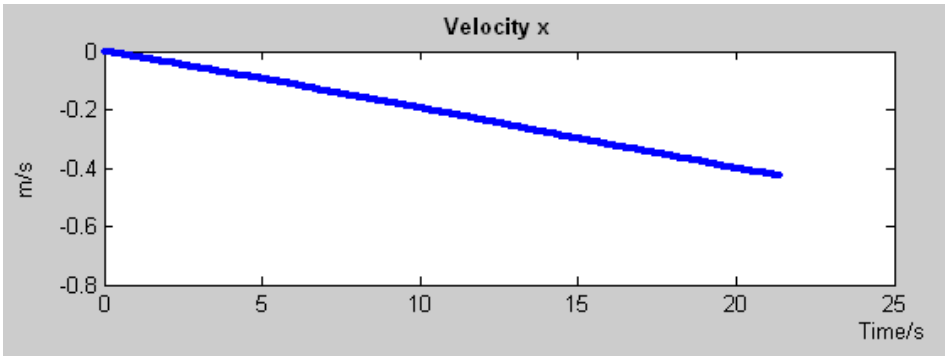
Figure 3.2(a) shows the drifts recorded from the RGA300CA IMU when it is stationary. Its shows a constant bias of approximately -0.021 m/s^2 and regular fluctuations of the signals even when there are no changes in acceleration. These errors are accumulated during the single integration process to get the velocity. Figure 3.2(b) shows a non zero velocity despite the IMU being stationary. Figure 3.2(c) shows that the errors from the recorded acceleration are further amplified during the double integration process to get the distance.

3.2.2 Compass

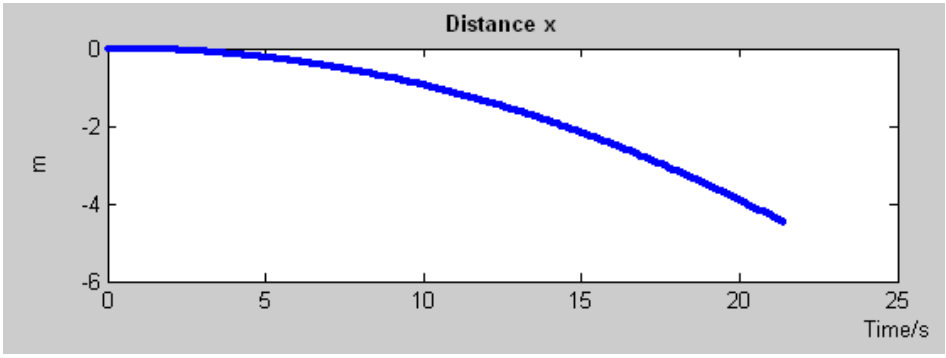
The compass is an instrument invented by the Chinese at around 2000 B.C. [28] for finding direction on Earth. The earliest compass is made up of a magnetized needle floating on water to allow it to freely pivot to align itself with the Earth's magnetic field. Technically, a compass is a magnetic device using a needle to indicate



(a)



(b)



(c)

Figure 3.2: (a) Drift of RGA300CA IMU along the x-axis when it is stationary. The signals change over time even when there are no changes in the acceleration. (b) The stationary IMU shows non zero velocity due to accumulated errors from the single integration process. (c) The stationary IMU shows non zero distance traveled due to accumulated errors from the double integration process.

the direction of the magnetic north of Earth's magnetosphere.

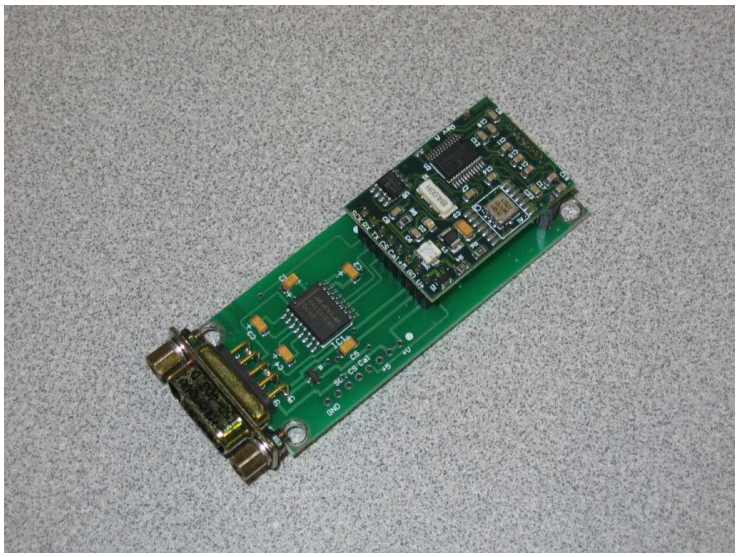


Figure 3.3: HMR3300 Digital compass from Honeywell.

Many variations of compasses have been created since its discovery by the Chinese. They include gyrocompasses, electronics fluxgate compasses, Hall-effect compasses, magnetoresistive and bearing compasses etc. Figure 3.3 shows an example of a magnetoresistive compass HMR3300 manufactured by Honeywell[3]. This compass consists of three axis of magnetoresistive sensors on board for sensing direction and an accelerometer to provide tilt (pitch and roll) sensing relative to the board's horizontal position [29].

[3]Company webpage: http://www.magneticsensors.com/

The main advantage of using the compass is that it requires no modification to the environment. However, the major drawback of compass is that its accuracy is easily affected by magnetic or ferromagnetic objects that are in its vicinity. As a result, compasses are usually not used for indoor mobile robot localization. This is because the steel structures in buildings are ferromagnetic objects that will cause large interference to the compass readings.

3.2.3 Global Positioning System

The global positioning system (GPS) is an active beacon system that was first developed by the United States Department of Defense and officially named NAVSTAR GPS [2, 25, 28]. The system was first developed for solely military uses but has already been made available for civilian uses.

The GPS system is made up a constellation of at least 24 satellites orbiting around Earth at a height of about 10,900 nautical miles[4]. The orbits of these satellites are arranged in a way such that at least six satellites are within line-of-sight from any locations on Earth. Each satellite makes two complete orbits each sidereal day[5] hence it passes over the same location on Earth once each day. This makes it possible to compute the exact location of these satellites with the knowledge of the time information from the atomic clocks carried by these satellites.

[4] 1 nautical mile = 1.852 km

[5] 1 sidereal day = 23.9344696 hours

A GPS receiver receives the time information of when the RF signals have been transmitted from the satellite. The difference between the time when a RF was transmitted and the time recorded by the internal clock of the GPS receiver when it has received the RF signal is the time-of-flight for the RF signal to travel from the satellite to the receiver. The range between the GPS receiver and satellite is then computed from the time-of-flight for the RF signal to travel from the satellite to the receiver. The location of the satellite is also computed from the time information. Based on the ranges of the receiver to at least three of the satellites and the locations of these satellites, the exact location of the GPS receiver is computed by trilateration techniques. Information from a fourth satellite is also used to compensate for any time errors between the GPS receiver and the satellites. Note that the location of the GPS receiver is given in the longitude, latitude and altitude format.

Figure 3.4: GPS from RF Solutions Ltd.

Figure 3.4 shows a GPS receiver module and antenna manufactured by RF Solutions Ltd[6]. This GPS module gives a position estimate that has an accuracy of within 5m and 50 percent circular error probability (CEP). The advantages of GPS for mobile robot localization is that it is low cost and easy to use. A receiver module and an antenna are all that is needed to receive location information from the satellites. The major drawback of the GPS system is its dependence on RF signals. The RF signals transmitted from the satellites will get attenuated by buildings, tree canopies, clouds and rain etc. This restricts GPS usage to only outdoor environments where there is no tree canopies, clouds or rain.

3.2.4 Odometer

Odometer refers to device that provides pose estimation of a wheeled robot by counting the number of wheel revolutions. Let n_{left} and n_{right} denotes the number of revolutions taken by the left and right wheel of the robot respectively. The linear distance traveled by the wheels is therefore given by

$$d_i = n_i \pi D, \quad \text{where i = left or right} \tag{3.1}$$

D is the diameter of the wheels. In a differential wheeled robot, the state of the robot $\mathbf{x_t} = [x_t \ y_t \ \theta_t]^T$ at current time t is given by

[6]Company webpage: www.rfsolutions.co.uk

$$\theta_t = \theta_{t-1} + \frac{d_{right} - d_{left}}{L_{axle}} \tag{3.2}$$

$$x_t = x_{t-1} + 0.5(d_{left} + d_{right}) \cos\theta_t \tag{3.3}$$

$$y_t = y_{t-1} + 0.5(d_{left} + d_{right}) \sin\theta_t \tag{3.4}$$

where L_{axle} is the distance between the floor contact points of the left and right wheels.

Figure 3.5: An 8 degree/step bipolar stepper motor from Shinano Kenshi Co. Ltd to drive the ER2 robot.

Many devices can be used as an odometer for counting the number of revolutions taken by the robot wheels. They include potentiometers, synchros, revolvers, encoders and tracking the control signals sent to stepper motors [2]. Figure 3.5 shows an 8 degree/step bipolar stepper motor from Shinano Kenshi Co. Ltd Japan[7]. This

[7]Company webpage: http://www.skcj.co.jp/english/indexe.html

stepper motor is used to drive the ER2 mobile robot. The stepper motor is driven by a regulated pulse train [30]. Each pulse drives a step (8^o) of the motor and the speed of the motor speed is determined by the frequency of the pulse train. The higher the frequency, the higher the speed. n_{left} and n_{right} from Equation 3.1 can be determined by counting the number of pulses used to drive the left and right motors respectively. The current state of the robot $\mathbf{x_t}$ can thus be determined from Equations 3.2 to 3.4.

The main advantages of odometers for mobile robot localization are the ease of implementation, low cost and it does not require any modifications to the environment. Odometer is therefore the most popular choice for robot localization. The major drawback is that it suffers from random errors caused by wheel slippage etc and systematic errors caused by unequal wheel diameters etc. These errors are accumulated as the robot travels a greater distance and eventually will grow unbounded if left unchecked.

3.2.5 Cricket Motes

Figure 3.6 shows the Cricket Motes designed by the MIT Computer Science and Artificial Intelligence Laboratory[8]. It is an indoor active beacon localization system. Each Cricket Mote shown in Figure 3.6 can be configured to work as beacon or listener and each beacon can also be configured to transmit an unique ID to the listener[7].

To set up an active beacon system, multiple beacons with unique IDs are attached to the ceiling at known position coordinates with respect to a fixed reference frame and a listener is attached to the robot. The beacons are capable of transmitting

[8]Laboratory webpage: http://cricket.csail.mit.edu/

Figure 3.6: Cricket Motes from MIT computer science and artificial intelligence laboratory.

both RF and ultrasonic signals to the listener. Each beacon periodically broadcasts it unique ID via RF signals and simultaneously broadcasts an ultrasonic pulse. The listener on the robot will receive the RF message and ultrasonic pulse if it is within the line-of-sight to the beacon.

Since RF travels about 10^6 times faster than ultrasound, the listener can use the time difference of arrival between the start of the RF message from a beacon and the corresponding ultrasonic pulse to infer its distance from the beacon. Problems from cross-talks of the ultrasonic pulses and the solutions to solve these problems are described in [31]. The robot is given a prior knowledge of the beacon position coordinates and their respective IDs. Thus the listener on the robot will be able to deduce the position coordinates of a beacon when it has received the ID information from its

RF message. The robot then compute its own position coordinates with respect to the fixed frame by trilateration techniques, based on its distances from three or more beacons and position coordinates of these beacons.

The Cricket Motes serves as a good alternative for the GPS as an active beacon system for indoor mobile robot localization. It has an excellent distance accuracy in the order of 1cm at a distance up to 3.5m and 2cm in the rest of the 10.5m range. One of its major drawback is that it requires a cumbersome procedure of attaching the beacons onto the ceilings and to obtain their precise position coordinates with respect to a fixed reference frame. A large number of beacons are also needed to cover large indoor environments and this means high cost. Figure 3.7 shows another major drawback for the Cricket Motes. The RF and ultrasonic signals transmitted from the beacons will get attenuated by obstacles that intercept the line-of-sight from the beacon to listener. This means that there will be blind spots where the robot is not able see three or more beacons to localize itself properly.

3.2.6 NorthStar Localization Kit

Figure 3.8 shows the NorthStar localization kit from Evolution Robotics, Inc[9]. The NorthStar is an indoor active beacon localization system where modulated IR light spots are used as uniquely identified landmarks by an advanced IR detector to determine relative position and heading [8, 9].

[9]Company webpage: http://www.evolution.com/

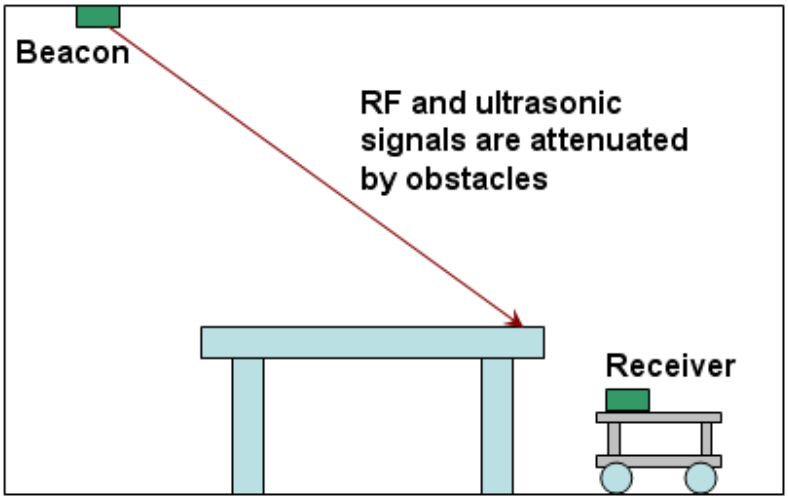


Figure 3.7: RF and ultrasonic signals from beacons may get attenuated by obstacles that intercept the line-of-sight from the beacon to receiver.

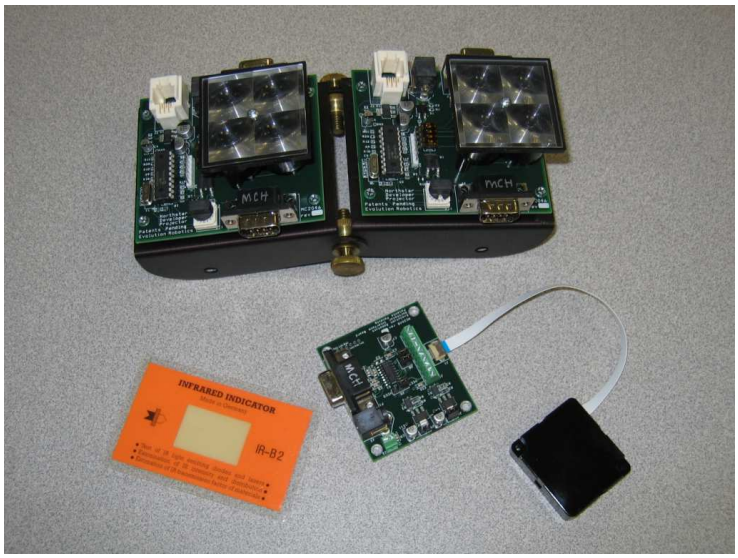


Figure 3.8: NorthStar projector kit (top), detector kit (bottom right) and infrared indicator (bottom left) from Evolution Robotics, Inc. The infrared indicator is used to detect IR light spots since they are not visible to naked eyes.

The NorthStar system consists of two basic components: Projector and Detector. Each NorthStar projector emits two modulated IR light spots that can be decoded by the detector. The NorthStar detector is a compact IR sensors equipped with onboard signal processing and a communication interface. It is used to track the distances and orientations to the IR light spots. The detector is also able to distinguish each IR light spot based on information from the modulated IR source.

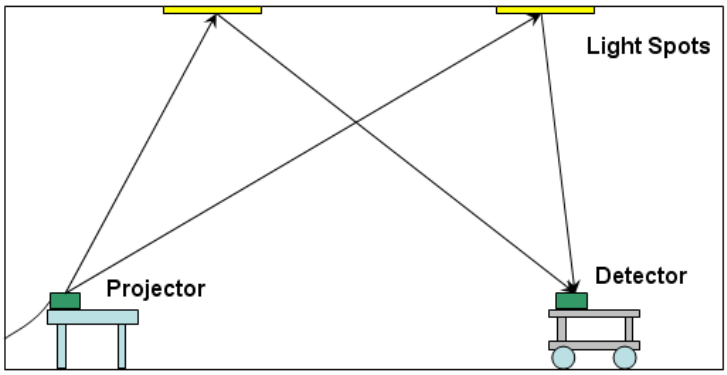


Figure 3.9: An illustration to show the setup of the NorthStar localization kit.

Figure 3.9 shows an illustration of a typical setup of the NorthStar localization kit. The projector is place at a fixed location and it projects two IR light spots with unique ID onto the ceiling at known locations with respect to a fixed reference frame. The detector is fixed onto the robot and it will track the distances and orientations of the IR light spots from the robot. The detector also track the unique ID of the IR light spots and hence their locations with respect to the fixed reference frame will be known. With the information of the distances and orientations of the IR light spots

from the robot and the locations of these light spots with respect to the fixed reference frame, the robot is able to compute its pose with respect to the fixed reference frame based on the two beacons triangulation techniques.

Figure 3.10 shows an illustration of triangulation with two beacons. The unique solution for the robot pose can only be obtained if the distances (i.e. d_1 and d_2) and orientations (i.e. α_1 and α_2) from the robot to the beacons are known. The solution will be ambiguous if only the distances (i.e. d_1 and d_2) from the beacons to the robot are known.

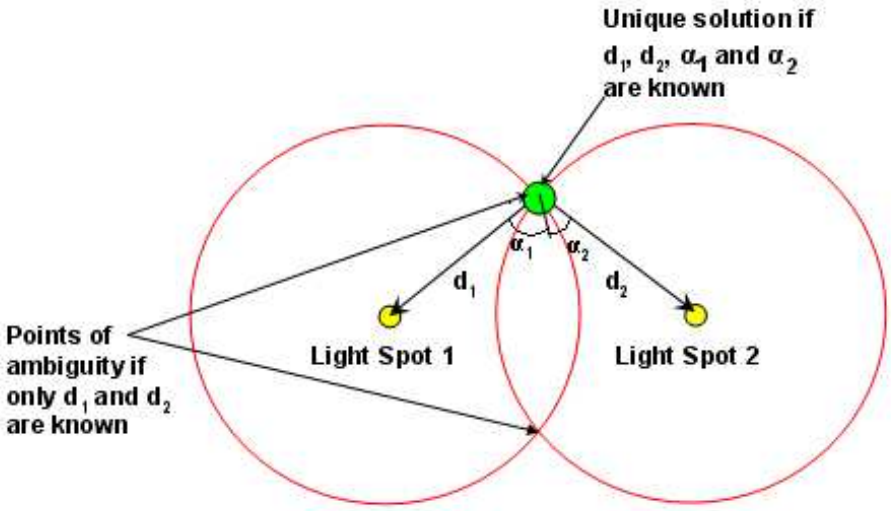


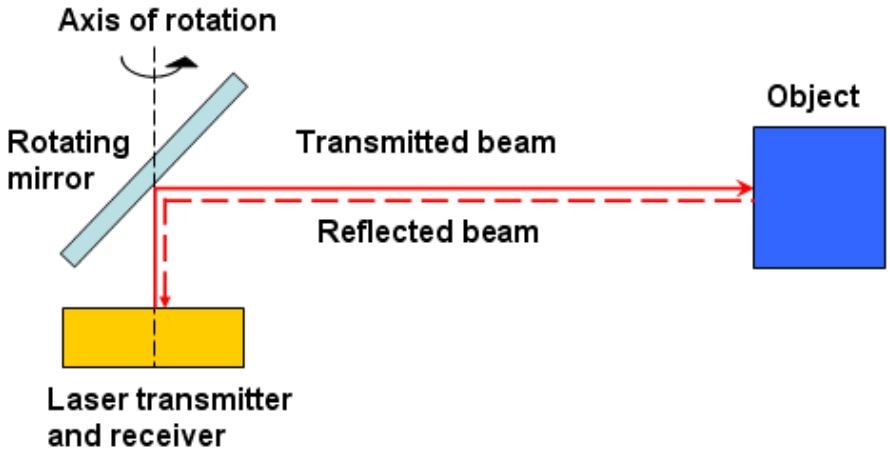
Figure 3.10: Triangulation with two beacons.

The NorthStar localization kit provides a good alternative to the GPS as an active beacon system for indoor mobile robot localization. However, there are several

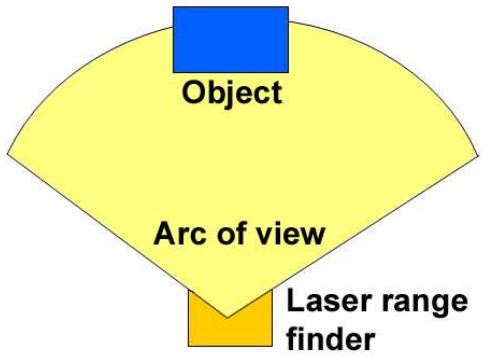
drawbacks. First, the IR light spots must be projected onto flat ceilings for efficient localization. This is not practical because many ceilings are designed with infrastructures such as beams which will cause disruptions to the IR light spots. Second, lightings on the ceilings will cause inaccuracies to the pose estimates due to interferences to the IR light spots. Third, at least two IR light spots must be detected for the robot to localize itself. The maximum range for the IR light from the projector to the ceiling is about 4m. This means that a large number of projectors are needed for large environments and hence increasing the cost. Fourth, the NorthStar localization kit requires modifications to be done in the environment. Lastly, similar to the Cricket motes, blind spots are created if there are obstacles that intercept the line-of-sight of the detector to the IR light spots.

3.2.7 Laser Range Finder

A laser range finder is a device that uses a laser beam to determine its range to a reflective object. A laser beam pulse is transmitted periodically from the laser range finder. The laser beam pulse gets reflected back to the laser range finder if it hits a reflective object. The distance from the laser range finder to the object is then computed based on the time-of-flight for the laser beam to travel to the object and back again. The laser range finders usually have an internal rotating mirror to deflect the laser beam pulse so that range measurements can be obtained over a certain arc of view. The rotation step angle of the mirror will determine the resolution of the laser range finder. Figure 3.11(a) shows an illustration of the internal rotating mirror. Figure 3.11(b) shows the plan view of Figure 3.11(a) which reveals the arc of view of



(a)



(b)

Figure 3.11: (a) Laser range finder with internal rotating mirror. (b) Plan view of (a) which shows the laser arc of view.

the laser range finder.

Figure 3.12 shows the URG-04LX laser range finder from Hokuyo Automatic Co. Ltd[10]. This laser range finder has an arc of view of resolution of $\pm 90^o$ with respect to its heading and a resolution of approximately 0.00612 rad [32]. This means that there

[10]Company webpage: http://www.hokuyo-aut.jp/

are 513 readings in every range scan from the URG laser range finder. The maximum range of this sensor is approximately 4095mm and its accuracy is experimentally found to be ±50mm with 68 percent of confidence (see Section 4.3.2).

Figure 3.12: URG-04LX laser range finder from Hokuyo Automatic Co. Ltd.

One of the advantages of laser range finder is that it is able to sense long range with high accuracy. In addition, it does not require any modifications of the environment. The laser range finder can be used both indoor and outdoor. The major drawback of laser range finder is that it can only detect objects that are within its two-dimensional scanning plane. This means objects that lie beyond the two-dimensional scanning plane will not be detected. Figure 3.13 shows an example of an undetected object that lies beyond the two-dimensional scanning plane of the laser sensor. Another drawback is that limited information are provided by the laser scan readings. It only tells the robot its distance relative to an object and this piece of information is often not sufficient to localize the robot. However, this problem can be overcome easily by doing feature extraction [11, 33] or building an occupancy grid map (see

Section 5.3) from the laser range scans.

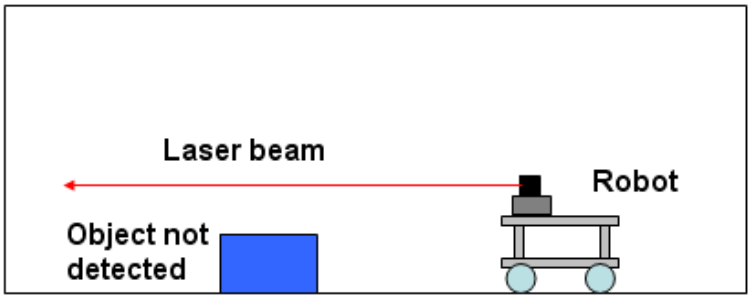


Figure 3.13: An object that lies beyond the two-dimensional scanning plane of the laser range laser range finder will not be detected.

3.3 Selection of Sensor(s) for Mobile Robot Localization

All the evaluated sensors have their respective strengths and weaknesses in the context of mobile robot localization and each sensor is also suitable for use in different environments. In this section, the sensor(s) suitable for localization of a mobile robot in both known and unknown indoor environments shall be selected for implementation.

The objective of this research is to estimate the state $\mathbf{x_t}$ of a robot in an indoor environment. Hence, the selected sensor(s) must be able to work well in indoor environments. The robot state $\mathbf{x_t}$ estimation must also be done in both known (see Chapter 4) and unknown (SLAM problem, see Chapter 5) environments. As a result, the selected sensor(s) must not make any modifications to the environment since it

can be unknown. Figure 3.14 shows an evaluation of the sensors according to the two selection criteria. Three sensors: IMU, odometer and laser range finder are found to fulfil all the two selection criteria.

		IMU	Compass	GPS	Odometer	Cricket Motes	NorthStar Localization Kit	Laser Range Finder
1.	Work Indoor	Y	N	N	Y	Y	Y	Y
2.	No modification to environment	Y	Y	Y	Y	N	N	Y

Figure 3.14: IMU, odometer and laser range finder are the three sensors that work indoor and requires no modifications to the environment.

It was mentioned in the previous chapter that the robot state $\mathbf{x_t}$ is estimated recursively from its previous state $\mathbf{x_{t-1}}$, current control data $\mathbf{u_t}$ and sensor measurement data $\mathbf{z_t}$. Both IMU and odometer provide information about the change in the robot state and can be used as $\mathbf{u_t}$. However, odometer is chosen since the cost of IMU is significantly higher than odometer. In addition, odometer is already available on the ER2 robot.

The laser range finder is used as $\mathbf{z_t}$ because it provides information about the environment of the robot, in contrast to odometer which provides information about the internal state of the robot.

CHAPTER 4

LOCALIZATION IN A KNOWN ENVIRONMENT

4.1 Introduction

Localization of a mobile robot in a known environment refers to the problem of determining the pose of the mobile robot relative to a given map of the environment. In other words, mobile robot localization problem can also be seen as a problem of coordinate transformation. The mobile robot has to establish correspondence between the global frame that is fixed onto the given map and its own local frame.

Localization problems in a known environment are generally classified into three groups according to their level of difficulties [10]. They are local localization, global localization and kidnapped problem in ascending level of difficulty. In local localization, the robot has to keep track of its pose from an initially known location in the map. In global localization, the robot is placed in an initially unknown location and the goal is to locate its pose within the map. The kidnapped problem is an extension to the global localization problem. The robot may get kidnapped and teleported to some other location within the map during the global localization operation. The robot has to detect its incorrect pose in the event of being kidnapped and locate its correct pose as soon as possible. The kidnapped problem is usually used to test the

robot's ability to recover from catastrophic localization failures.

In this chapter, some of the probabilistic approaches to solve the localization problem in a known environment will be first discussed. Next, the details of the particle filter which will be used to solve all the three localization problems will be presented. Finally, the simulation and implementation results of the particle filter for all the localization problems in known environment will be shown.

4.2 Related Works

Some of the related works that uses probabilistic approach to solve the localization problem in a known environment will be discussed in this section.

4.2.1 Localization with Extended Kalman Filter

The *extended Kalman filter* (EKF) [1, 10, 11, 34] is perhaps the most established algorithm to implement the Bayes filter for the localization of mobile robots because of its robustness and efficiency. The EKF algorithm is a recursive method of estimating the pose of the robot with noisy sensor readings.

A key feature of the EKF is that it maintains a posterior belief $bel(\mathbf{x_t})$ of the pose estimate, which follows a Gaussian distribution, represented by a mean $\mathbf{x_t}$ and covariance $\mathbf{P_t}$. The mean $\mathbf{x_t}$ represents the most likely pose of the robot at time t and covariance $\mathbf{P_t}$ represents the error covariance of this estimate. The EKF consists of two steps: the prediction and update step. In the prediction step, the predicted belief

$\overline{bel}(\mathbf{x_t})$ is first computed using a motion model which describes the state dynamics of the robot. $\overline{bel}(\mathbf{x_t})$ is subsequently transformed into $bel(\mathbf{x_t})$ by incorporating the sensor measurements in the update step.

As mentioned above, the predicted belief $\overline{bel}(\mathbf{x_t})$, which is represented by the predicted mean $\overline{\mathbf{x}}_\mathbf{t}$ and covariance $\overline{\mathbf{P}}_\mathbf{t}$, is computed from the prediction step given by

$$\overline{\mathbf{x}}_\mathbf{t} = \mathbf{f}(\mathbf{x_{t-1}},\ \mathbf{u_t}) \tag{4.1}$$

$$\overline{\mathbf{P}}_\mathbf{t} = \mathbf{F_t}\ \mathbf{P_{t-1}}\ \mathbf{F_t}^T + \mathbf{Q_t} \tag{4.2}$$

where $\mathbf{f}(.)$ is the motion model of the mobile robot, $\mathbf{F}$ is the Jacobian of $\mathbf{f}(.)$ evaluated at $\mathbf{x_{t-1}}$, $\mathbf{Q_t}$ is the error covariance of the motion model and $\mathbf{u_t}$ is the control data of the robot.

The predicted belief $\overline{bel}(\mathbf{x_t})$ is subsequently transformed into the desired belief $bel(\mathbf{x_t})$ by incorporating the sensor measurement $\mathbf{z_t}$ in the update step shown in Equations 4.3, 4.4 and 4.5.

$$\mathbf{K_t} = \overline{\mathbf{P}}_\mathbf{t}\mathbf{H_t}^T(\mathbf{H_t}\overline{\mathbf{P}}_\mathbf{t}\mathbf{H_t}^T + \mathbf{R_t})^{-1} \tag{4.3}$$

$$\mathbf{x_t} = \overline{\mathbf{x}}_\mathbf{t} + \mathbf{K_t}(\mathbf{z_t} - \mathbf{h}(\overline{\mathbf{x}}_\mathbf{t}, \vartheta)) \tag{4.4}$$

$$\mathbf{P_t} = (\mathbf{I} - \mathbf{K_t}\mathbf{H_t})\overline{\mathbf{P}}_\mathbf{t} \tag{4.5}$$

$\mathbf{K_t}$ computed in Equation 4.3 is called the *Kalman gain*. It specifies the degree to which the measurement $\mathbf{z_t}$ should be incorporated into the new state estimate. Equation 4.4 computes the mean $\mathbf{x_t}$ by adjusting it in proportion to the Kalman gain

$\mathbf{K_t}$ and the deviation of the actual measurement $\mathbf{z_t}$ with the predicted measurement $\mathbf{h}(\overline{\mathbf{x}}_\mathbf{t}, \vartheta)$. It is important to note that the sensor measurement $\mathbf{z_t} = [z_t^1 \ z_t^2 ...]^T$ refers to coordinates of a set of observed features instead of the raw sensor readings. Many feature extraction algorithms [11, 33, 35] are available to extract features from the raw sensor readings. The sensor measurement model $\mathbf{h}(.)$ gives the predicted measurement from the given feature-based map ϑ [36, 37, 38] and predicted mean $\overline{\mathbf{x}}_\mathbf{t}$. $\mathbf{H_t}$ is the Jacobian of $\mathbf{h}(.)$ evaluated at $\mathbf{x_{t-1}}$. $\mathbf{R_t}$ is the error covariance of the sensor measurement model. Finally, the covariance $\mathbf{P_t}$ of the posterior belief $bel(\mathbf{x_t})$ is computed in Equation 4.5 by adjusting for the information gain resulting from the measurement.

4.2.2 Localization with Correlation

There exists a number of algorithms that do robot localization with correlation [12, 13, 39]. Typically, these algorithms first compile a small number of consecutive sensor readings into a local map denoted by ϑ_{local}. Next, the local map ϑ_{local} is compared with the given global map at all possible poses of the robot. Note that the local map ϑ_{local} is usually built with respect to the robot frame. Therefore, the local map ϑ_{local} has to be transformed into the reference frame of the global map prior to comparison. A correlation value for each comparison will be computed. The more similar the local map ϑ_{local} and global map at the possible pose, the higher the correlation value. The pose in the global map that yields the highest correlation value with the local map ϑ_{local} is taken as the estimated robot pose.

4.2.3 Localization with Particle Filter

In recent years, there is an increasing interest in the use of particle filter for robot localization [10, 14, 15, 16, 17]. The intuition behind the particle filter is to represent the posterior belief $bel(\mathbf{x_t})$ by a finite sample set of M weighted particles drawn according to this distribution. Similar to the EKF, the particle filter consists of the prediction and update steps. In the prediction step, samples of the particles are drawn from a motion model of the robot to represent the predicted belief $\overline{bel}(\mathbf{x_t})$. The particles are then weighted according to the sensor measurements in the update step. Finally, the predicted belief $\overline{bel}(\mathbf{x_t})$ is transformed into the posterior belief $bel(\mathbf{x_t})$ by resampling the particles according to their weights.

The increasing interest in particle filter is due to several reasons. First, raw sensor measurements of the environment are used in particle filter localization. This is unlike EFK localization which requires feature extraction and correlation localization which requires local mapping. Second, the particle filter is non-parametric. This means that the particle filter is more robust than EKF because it does not assume Gaussian posterior belief distributions $bel(\mathbf{x_t})$. Third, different variations of the particle filter is capable of solving all the three problems of localization in a known environment. Fourth, the particle filter is easy to implement. Unlike the EKF, there is no need to derive complicated Jacobians for the particle filter.

Due to the advantages of the particle filter over other the localization algorithms, it shall be investigated in more detail and implemented on the ER2 mobile robot for

localization in this thesis.

4.3 Method Investigated - The Particle Filter

It was mentioned in Section 2.2.3 that it is not possible to represent belief distribution in a digital computer because the continuous state space $\mathbf{x_t}$. The particle filter attempts to overcome this problem by representing the belief distribution $bel(\mathbf{x_t})$ by a finite set of random state samples drawn from this distribution.

In particle filter, the belief distribution $bel(\mathbf{x_t})$ is represented by a finite sample set of particles denoted by

$$\xi_t := \chi_t^{[1]}, \chi_t^{[2]}, ..., \chi_t^{[M]} \tag{4.6}$$

where $\chi_t^{[m]} = [\mathbf{x_t^{[m]}}\ w^{[m]}]^T$ denotes the m^{th} particle. Here, $\mathbf{x_t^{[m]}}$ is a random variable that represents the hypothesized state of the m^{th} particle and w_t is a non-negative value called the importance factor which determines the weight of each particle.

Table 4.1 shows the pseudo code for the particle filter algorithm. The inputs to the algorithm are the previous particle set $\xi_{\mathbf{t-1}}$, the most recent control data $\mathbf{u_t}$ and sensor measurement $\mathbf{z_t}$. The particle filter algorithm first generates a temporary particle set $\overline{\xi}_{\mathbf{t}}$ that represents the predicted belief distribution $\overline{bel}(\mathbf{x_t})$ in the prediction step. It is then followed by the update step that transforms the predicted belief distribution $\overline{bel}(\mathbf{x_t})$ into the posterior belief distribution $bel(\mathbf{x_t})$. In details:

1: **Particle_filter**$(\xi_{\mathbf{t-1}}, \mathbf{u_t}, \mathbf{z_t})$:
2: $\overline{\xi}_{\mathbf{t}} = \xi_{\mathbf{t}} = \emptyset$
3: **for** $m = 1$ *to* M **do**
4: *sample* $\mathbf{x}_{\mathbf{t}}^{[\mathbf{m}]} \sim p(\mathbf{x_t} \mid \mathbf{u_t},\ \mathbf{x}_{\mathbf{t-1}}^{[\mathbf{m}]})$
5: $w_t^{[m]} = p(\mathbf{z_t} \mid \mathbf{x}_{\mathbf{t}}^{[\mathbf{m}]},\ \vartheta)$
6: $\overline{\chi}_{\mathbf{t}}{}^{[m]} = [\mathbf{x}_{\mathbf{t}}^{[\mathbf{m}]}\ w_t^{[m]}]^T$
7: **end for**
8: $\xi_{\mathbf{t}} = resample(\overline{\xi}_{\mathbf{t}})$
9: **Return** $\xi_{\mathbf{t}}$

Table 4.1: The particle filter algorithm.

1. **Prediction**: Line 4 of the algorithm generates the hypothetical state $\mathbf{x}_{\mathbf{t}}^{[\mathbf{m}]}$ by sampling from the state transition probability distribution $p(\mathbf{x_t} \mid \mathbf{u_t},\ \mathbf{x}_{\mathbf{t-1}}^{[\mathbf{m}]})$. The state transition probability $p(\mathbf{x_t} \mid \mathbf{u_t},\ \mathbf{x}_{\mathbf{t-1}}^{[\mathbf{m}]})$ is obtained from the odometry motion model. See Section 4.3.1 for more details on the implementation of the odometry motion model. The set of particles obtained after M iterations is the discrete representation of the predicted belief $\overline{bel}(\mathbf{x_t})$.

2. **Update**: The update step of the particle filter algorithm consists of two steps: importance factor and resampling.

   (a) **Importance Factor**: The importance factor $w_t^{[m]}$ for the m^{th} particle at time t is computed in Line 5 of the algorithm. Importance factors are used to incorporate the measurement $\mathbf{z_t}$ into the particle set and the importance factor of the m^{th} particle is given by the measurement probability $p(\mathbf{z_t} \mid \mathbf{x}_{\mathbf{t}}^{[\mathbf{m}]},\ \vartheta)$. ϑ represents the given map of the environment that the mobile robot is working in. The measurement probability is computed from the sensor measurement model. See Section 4.3.2 for more details

on computing the measurement probability. It should be noted that the particles with hypothetical states closer to the posterior belief distribution $bel(\mathbf{x_t})$ will have a higher importance factor.

(b) **Resampling**: The resampling step in Line 8 of the algorithm is perhaps the most important part of the particle filter. Resampling draws with replacement M particles from the temporary set $\overline{\xi}_\mathbf{t}$. The probability of drawing each particles is given by its importance weight. This means that the particles with higher importance weight (also means that the hypothetical states of these particle are closer to the posterior belief distribution $bel(\mathbf{x_t})$) will have a higher chance of appearing in $\xi_\mathbf{t}$. Consequently, the particles will be approximately distributed according to the posterior belief distribution $bel(\mathbf{x_t}) = \eta\ p(\mathbf{z_t} \mid \mathbf{x_t})\ \overline{bel}(\mathbf{x_t})$ after the resampling step. See Section 4.3.3 for more details on the resampling algorithm.

4.3.1 Odometry Motion Model

The distribution of the state transition probability $p(\mathbf{x_t} \mid \mathbf{u_t},\ \mathbf{x}^{[\mathbf{m}]}_{\mathbf{t-1}})$ in the particle filter algorithm is computed from the odometry motion model. It describes the posterior distribution over the kinematic states that a robot assumes when executing the control action at $\mathbf{x}^{[\mathbf{m}]}_{\mathbf{t-1}}$. Information of the control action is provided by the control data $\mathbf{u_t}$. Note that odomtery is used to compute the state transition probability $p(\mathbf{x_t} \mid \mathbf{u_t},\ \mathbf{x}^{[\mathbf{m}]}_{\mathbf{t-1}})$ because it was selected from the sensors evaluation in the previous chapter.

The prediction step in Line 4 of the particle filter algorithm seeks to generate a random $\mathbf{x_t^{[m]}}$ from the motion model $p(\mathbf{x_t} \mid \mathbf{u_t}, \mathbf{x_{t-1}^{[m]}})$. Table 4.2 shows the pseudo code for generating a random sample from the odometry motion model. The inputs are the current control data $\mathbf{u_t}$ and a hypothetical state $\mathbf{x_{t-1}^{[m]}}$ of the robot at $t-1$.

The relative motion information of the robot in the time interval $(t-1, t]$ can be represented by a translated distance δ_{trans} and a rotated angle δ_{rot}. Lines 2 and 3 of the algorithm computes δ_{trans} and δ_{rot} from the previous state of the robot $\mathbf{x_{t-1}^{[m]}}$ and the control data $\mathbf{u_t}$. δ_{trans} and δ_{rot} are taken to be corrupted by noise. This is because the control data $\mathbf{u_t}$ is obtained from odometry readings that are corrupted by noise.

Lines 4 to 16 of the algorithm assumed that the "true" values of the translation and rotation denoted by $\widehat{\delta}_{trans}$ and $\widehat{\delta}_{rot}$ are obtained from δ_{trans} and δ_{rot} by subtracting independent Gaussian noise with zero mean and standard deviation denoted by σ. The $if-else$ conditions imposed from Lines 4 to 16 of the algorithm are to ensure that no random noise is subtracted from δ_{trans} and δ_{rot} if there is no motion.

Note that σ_1, σ_2, σ_3 and σ_4 do not have the same value. σ_1 denotes the standard deviation when there is only pure rotation. σ_2 denotes the standard deviation when there is only pure translation. σ_3 and σ_4 denotes the translational and rotational standard deviations when there are both translation and rotation. The values for the standard deviation have to be determined experimentally. First, the robot is made to perform pure translation, pure rotation, as well as translation and rotation over a

1: **sample_motion_model_odometry**$(\mathbf{u_t}, \mathbf{x_{t-1}^{[m]}})$:

2: $\delta_{trans} = \sqrt{(x_t - x_{t-1}^{[m]})^2 + (y_t - y_{t-1}^{[m]})^2}$
3: $\delta_{rot} = \theta_t - \theta_{t-1}^{[m]}$

4: **if** δ_{trans} is $\emptyset$ and δ_{rot} is $\emptyset$ **then**
5: $\quad \widehat{\delta}_{trans} = \delta_{trans}$
6: $\quad \widehat{\delta}_{rot} \quad = \delta_{rot}$
7: **else if** δ_{trans} is $\emptyset$ and δ_{rot} not $\emptyset$ **then**
8: $\quad \widehat{\delta}_{trans} = \delta_{trans}$
9: $\quad \widehat{\delta}_{rot} \quad = \delta_{rot} - \mathbf{sample}(\sigma_1)$
10: **else if** δ_{trans} not $\emptyset$ and δ_{rot} is $\emptyset$ **then**
11: $\quad \widehat{\delta}_{trans} = \delta_{trans} - \mathbf{sample}(\sigma_2)$
12: $\quad \widehat{\delta}_{rot} \quad = \delta_{rot}$
13: **else**
14: $\quad \widehat{\delta}_{trans} = \delta_{trans} - \mathbf{sample}(\sigma_3)$
15: $\quad \widehat{\delta}_{rot} \quad = \delta_{rot} - \mathbf{sample}(\sigma_4)$
16: **end if**

17: x $= x_{t-1}^{[m]} + \widehat{\delta}_{trans}\ cos(\theta_{t-1}^{[m]} + \widehat{\delta}_{rot})$
18: y $= y_{t-1}^{[m]} + \widehat{\delta}_{trans}\ sin(\theta_{t-1}^{[m]} + \widehat{\delta}_{rot})$
19: $\theta = \theta_{t-1}^{[m]} + \widehat{\delta}_{rot}$

20: **return** $\mathbf{x_t^{[m]}} = [x\ y\ \theta]^T$

Table 4.2: Algorithm for sampling from the odometry motion model.

certain fixed interval given by $\mathbf{u_t}$. Next, the "true" values of the respective motions are measured physically. This is done N times so that N "true" values can be obtained for each of the motions over the fixed interval given by $\mathbf{u_t}$. A histogram of the frequency distribution for the "true" values of each motion is then plotted and fitted with a Gaussian curve. Finally, σ_1, σ_2, σ_3 and σ_4 can be obtained from the histogram that represents the respective motion. For the ER2 robot used in this research, $\sigma_1 =$

5^o, $\sigma_2 = 8$mm, $\sigma_3 = 10$mm and $\sigma_4 = 7^o$. Note that the values for the standard deviation are obtained for a rotation interval of 50^o and translation interval of 1000mm.

Finally, in Lines 17 to 19 of the algorithm, the "true" pose $\mathbf{x_t^{[m]}} = [x\ y\ \theta]^T$ of the robot is computed from its initial pose $\mathbf{x_{t-1}^{[m]}}$ and the "true" translation and rotation using the state equations of the robot.

It is important to note that most programming compilers are only capable to generate random numbers that follow an uniform distribution. However, the odometry motion model in Table 4.2 requires random noises that follow normal distributions. Table 4.3 gives an algorithm to generate a random number that follows normal distribution with zero mean from random numbers that follow an uniform distribution [10]. The input to the algorithm is the desired standard deviation σ. **rand**(a,b) in Line 2 of the algorithm denotes a random number generator with uniform distribution in [a,b].

1: **sample_normal_distribution**
2: **return** $\frac{1}{2}\sum_{i=1}^{12}\mathbf{rand}(-\sigma,\ \sigma)$

Table 4.3: Algorithm for sampling from a normal distribution with zero mean and standard deviation σ.

4.3.2 Sensor Measurement Model

The sensor measurement probability $p(\mathbf{z_t} \mid \mathbf{x_t^{[m]}},\ \vartheta)$ in the particle filter algorithm is computed from the sensor measurement model. Unlike the odometry motion model which generates a sample from $p(\mathbf{x_t} \mid \mathbf{u_t},\ \mathbf{x_{t-1}^{[m]}})$, the sensor measurement model seeks to compute the probability value of $p(\mathbf{z_t} \mid \mathbf{x_t^{[m]}},\ \vartheta)$.

The URG laser range finder was selected to provide $\mathbf{z_t}$ from the sensors evaluation in the previous chapter. Hence, there are a total of 513 readings in one scan and the sensor measurement shall be denoted by

$$\mathbf{z_t} = [z_t^1 \quad z_t^2 \quad ... \quad z_t^K]^T, \quad \text{where K} = 513 \tag{4.7}$$

The probability value $p(z_t^k \mid \mathbf{x_t^{[m]}},\ \vartheta)$ under a single sensor measurement reading z_t^k is assumed to follow a normal distribution with mean z_t^{k*} and standard deviation σ_{hit}. This probability value is computed by

$$p(z_t^k \mid \mathbf{x_t^{[m]}},\ \vartheta) = \{\frac{1}{\sqrt{2\pi\sigma_{hit}^2}} e^{-\frac{1}{2}\frac{(z_t^k - z_t^{k*})^2}{\sigma_{hit}^2}}\} \tag{4.8}$$

The standard deviation σ_{hit} of the sensor measurement model has to be obtained experimentally. First, an object is placed at a fixed distance away from the laser range finder. Next, N readings are obtained from the laser range finder for this object. A histogram of the frequency distribution for these readings is then plotted and fitted with a Guassian curve. The standard deviation of the Gaussian curve is taken to be σ_{hit}. σ_{hit} is found to be 50mm for the URG laser range finder used in this research. The mean z_t^{k*} from Equation 4.8 is the predicted k^{th} sensor reading

from the hypothetical state $\mathbf{x_t^{[m]}}$ of the m^{th} particle and a given occupancy grid map ϑ of the environment. z_t^k is the k^{th} sensor reading from the laser range finder. Note that an occupancy grid map (see Section 5.3 for more details) is a representation of the environment as a tessellation of retangloid grid cells and each grid cell represents either occupied or unoccupied space in the environment.

Assuming that the noise in each sensor range reading are independent of each other, the total probability $p(\mathbf{z_t} \mid \mathbf{x_t^{[m]}},\ \vartheta)$ is therefore given by the product of the individual measurement likelihoods shown in Equation 4.9 where η is a normalizer to ensure that the probability stays within 0 to 1.

$$p(\mathbf{z_t} \mid \mathbf{x_t^{[m]}},\ \vartheta) = \eta \prod_{k=1}^{K} p(z_t^k \mid \mathbf{x_t^{[m]}},\ \vartheta), \quad \text{where K} = 513 \tag{4.9}$$

The probability value that is computed from Equation 4.9 can be seen as a measure of discrepancy between the laser scan measurements $\mathbf{z_t}$ and the predicted measurement readings

$$\mathbf{z_t^*} = [z_t^{1*}\ z_t^{2*}\ ...\ z_t^{K*}]^T, \quad \text{where K} = 513 \tag{4.10}$$

The higher the discrepancy between $\mathbf{z_t}$ and $\mathbf{z_t^*}$, the lower the probability value and hence less weight for the hypothetical state of the m^{th} particle. A particle with lesser weight implies that the likelihood of the particle depicting the true state is lower.

Ray Casting Algorithm

The predicted measurement $\mathbf{z_t^*}$ is computed from the *ray casting* algorithm [40]. The ray casting algorithm can be seen as a process of finding the sensor measurement data from an imaginary laser range finder attached to a hypothetical state $\mathbf{x_t^{[m]}}$. Imaginary "rays" are casted from this laser range finder into the environment. The environment must be represented as an occupancy grid map ϑ. Each imaginary "ray" is terminated at the point where it hits an obstacle or when its length exceeded the maximum range of the laser range finder. The length of these "rays" are subsequently taken as $\mathbf{z_t^*}$. Figure 4.1 shows an illustration of the k^{th} "ray" casted from an hypothetical state $\mathbf{x_t^{[m]}}$. The "ray" is terminated when it hits an obstacle and its length is taken to be z_t^{k*}.

Table 4.4 shows the pseudo code of the ray casting algorithm for the k^{th} "ray" casted from $\mathbf{x_t^{[m]}}$. The inputs to the ray casting algorithm are the hypothetical state $\mathbf{x_t^{[m]}}$, occupancy grid map ϑ of the environment and "ray" index k. The algorithm consists of three steps:

1. The grid coordinates of $\mathbf{x_t^{[m]}}$ denoted by $[i_{local}\ j_{local}]^T$ is computed.

2. The grid coordinates of the end point of the k^{th} "ray" projected from $\mathbf{x_t^{[m]}}$ in the event where there is no obstacle obstructions is computed. Let $[i_{beam}\ j_{beam}]^T$ denote this grid coordinates.

3. The predicted measurement z_t^{k*} is computed using the *Bresenham line algorithm* [40].

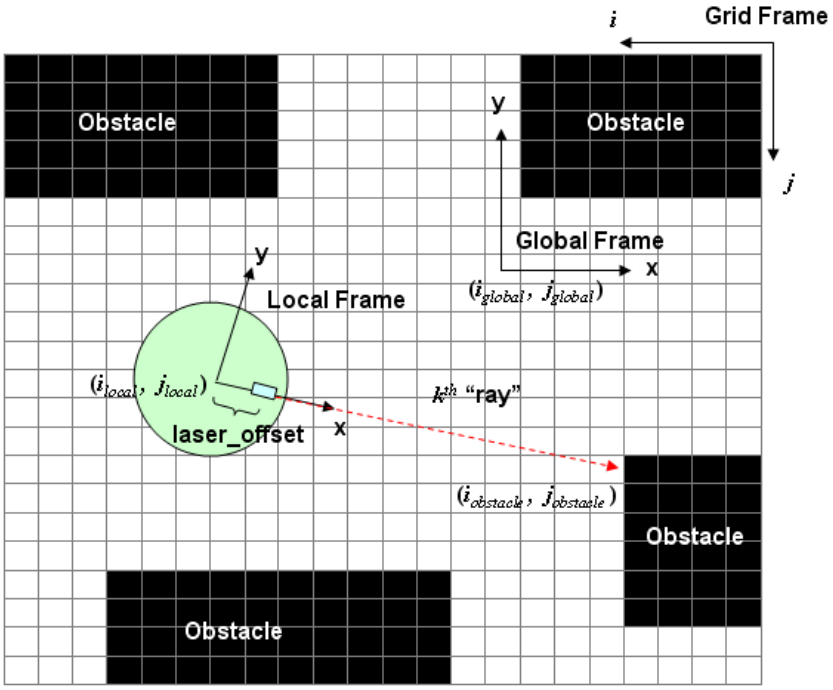


Figure 4.1: 2D ray casting from a hypothetical state $\mathbf{x_t^{[m]}}$.

In details: Lines 2 and 3 of the algorithm transform the coordinates of the hypothetical state $\mathbf{x_t^{[m]}}$ into grid cell coordinates $[i_{local}\ j_{local}]^T$. Note that $[i_{global}\ j_{global}]^T$ refers to the coordinates of the global fixed frame with respect to the grid frame and **grid_resolution** denotes the scale of the grid cell. For example, **grid_resolution** = 1m implies that each grid cell represents 1m^2 in the real world. It should also be noted that $[i_{global}\ j_{global}]^T$ and **grid_resolution** are constant values that are pre-determined. Figure 4.1 shows the relationship between the global fixed frame, grid frame and the local frame of the hypothetical state.

Lines 4 to 7 computes the grid coordinates $[i_{beam}\ j_{beam}]^T$ for the end point of the k^{th} "ray" projected from $\mathbf{x_t^{[m]}}$ in the event where there is no obstacle obstructions.

Note that $[i_{beam}\ j_{beam}]^T$ is computed in two steps. First, the end point coordinates $[x_{beam}\ y_{beam}]^T$ with respect to the global frame is computed in Lines 4 and 5. Next, $[i_{beam}\ j_{beam}]^T$ is computed from $[x_{beam}\ y_{beam}]^T$ by doing coordinates transformation in Lines 6 and 7. **laser_max_range** refers to the maximum range of the laser range finder. The maximum range of the URG laser range finder used in this research is 4095mm. **laser_resolution** refers to the angle between consecutive laser beams. The **laser_resolution** value for the URG laser range finder is approximately 0.00612 rad. The laser range finder is placed at 80mm away from the center of the robot along its x-axis. Hence, **laser_offset** = 80mm is needed to account for this offset in the computation of $[x_{beam}\ y_{beam}]^T$. Figure 4.1 shows the position of the laser range finder as seen from the hypothetical state $\mathbf{x_t^{[m]}}$.

Lines 8 to 19 ensures that $[i_{beam}\ j_{beam}]^T$ stay within the size of the given map. The size of the map ϑ is $[i_{min},\ i_{max}]$ in the i direction and $[j_{min},\ j_{max}]$ in the j direction. Finally, the predicted measurement z_t^{k*} is computed by the Bresenham line algorithm in Line 20.

Bresenham Line Algorithm

Table 4.5 shows the pseudo code of the Bresenham line algorithm. The inputs to the algorithm are $[i_{local}\ j_{local}]^T$ and $[i_{beam}\ j_{beam}]^T$ from the ray casting algorithm. The algorithm returns the predicted measurement z_t^{k*}, which is also the distance between $\mathbf{x_t^{[m]}}$ and the nearest obstacle in the direction of the k^{th} "ray". This is done by extrapolating a "ray" from $[i_{local}\ j_{loacl}]^T$ to $[i_{beam}\ j_{beam}]^T$ through the grid cells of the occupancy grid map. The distance between $[i_{local}\ j_{local}]^T$ and the first occupied grid

1: **ray_casting**($\mathbf{x_t^{[m]}}$, ϑ, k):

2: $i_{local} = i_{global} - x/$**grid_resolution**
3: $j_{local} = j_{global} - y/$**grid_resolution**

4: $x_{beam} = -$**laser_max_range** $* cos(\theta + k *$ **laser_resolution** $+ \frac{\pi}{4}) +$ **laser_offset** $*$ $cos(\theta) + x$
5: $y_{beam} = -$**laser_max_range** $* sin(\theta + k *$ **laser_resolution** $+ \frac{\pi}{4}) +$ **laser_offset** $*$ $sin(\theta) + y$
6: $i_{beam} = i_{global} - x_{beam}/$**grid_resolution**
7: $j_{beam} = j_{global} - y_{beam}/$**grid_resolution**

8: **if** $i_{beam} > i_{max}$ **then**
9: $\quad i_{beam} = i_{max}$
10: **end if**
11: **if** $i_{beam} < i_{min}$ **then**
12: $\quad i_{beam} = i_{min}$
13: **end if**
14: **if** $j_{beam} > j_{max}$ **then**
15: $\quad j_{beam} = j_{max}$
16: **end if**
17: **if** $j_{beam} < j_{min}$ **then**
18: $\quad j_{beam} = j_{min}$
19: **end if**

20: **return** $z_t^{k*} =$ **bresenham_line**$(i_{local},\ j_{local},\ i_{beam},\ j_{beam})$

Table 4.4: The ray casting algorithm for the k^{th} "ray" casted from $\mathbf{x_t^{[m]}}$.

cell denoted by $[i_{obstacle}\ j_{obstacle}]^T$ that obstructs the path of the "ray" is the predicted measurement on the map scale. The actual predicted measurement z_t^{k*} is obtained after accounting for the map scale.

The basic Bresenham line algorithm is only applicable to "rays" that are casted downward and to the right with a gradient between -1 and 0. A full implementation

of the algorithm requires the "rays" to be casted in all directions. Lines 2 to 6 allows the algorithm to work for "rays" with the magnitude of its gradient that is more than 1. This is achieved by reflecting the "ray", with the magnitude of its gradient more than 1, across the line y=x to change the magnitude of the gradient into lesser than 1. Lines 7 to 12 allows the algorithm to work for "rays" that are casted from right to left. This is easily achieved by swapping i_{local} with i_{beam} and j_{local} with j_{beam}.

The grid coordinates of the cells that cross the path of the "ray" denoted by $[i_{intercept}\ j_{intercept}]^T$ are compute from Lines 13 to 36. This is done by checking if $[i\ j]^T$ depicts a new cell each time i is increased by 1 and j is increased by j_{step}. The check is carried out from Lines 31 to 35. Note that Line 17 assigns j_{step} with -1 if the "ray" is casted upward. The grid cells $[i_{intercept}\ j_{intercept}]^T$ are checked for occupancy from Lines 21 to 30. Lines 21 to 25 checks for the first occupied cell along the casted "ray". In the case where $i_{local} > i_{beam}$, Lines 26 to 31 of the algorithm will check for the last occupied cell along the casted "ray". This is because the "ray" is casted backward when $i_{local} > i_{beam}$. Note that $\phi_{forward}$ is set to false if the "ray" is casted backward. The first or last occupied cell from the two cases are taken to be $[i_{obstacle}\ j_{obstacle}]^T$.

Finally, the Euclidean distance between $[i_{local}\ j_{local}]^T$ and $[i_{obstacle}\ j_{obstacle}]^T$ after accounting for the map scaling is taken to be the predicted measurement z_t^{k*}.

1: **bresenham_line**($i_{local},\ j_{local},\ i_{beam},\ j_{beam}$):

2: $steep = \mid j_{beam} - j_{local} \mid\ > \mid i_{beam} - i_{local} \mid$
3: **if** *steep is true* **then**
4: **swap**(i_{local}, j_{local})
5: **swap**(i_{beam}, j_{beam})
6: **end if**
7: $\phi_{forward} = true$
8: **if** $i_{local} > i_{beam}$ **then**
9: **swap**(i_{local}, i_{beam})
10: **swap**(j_{local}, j_{beam})
11: $\phi_{forward} = false$
12: **end if**

13: $\delta i = i_{beam} - i_{local}$
14: $\delta j = \mid j_{beam} - j_{local} \mid$
15: $error = 0$
16: $j = j_{local}$
17: **if** $j_{local} < j_{beam}$ **then** $j_{step} = 1$ **else** $j_{step} = -1$ **end if**

18: **for** $i = i_{local}$ *to* i_{beam} **do**
19: $(i_{intercept},\ j_{intercept}) \leftarrow (i,\ j)$
20: **if** *steep is true* **then swap**($i_{intercept},\ j_{intercept}$) **end if**
21: **if** $\phi_{forward}$ *is true* **then**
22: **if** ϑ *at* $(i_{intercept},\ j_{intercept})$ *is occupied* **then**
23: $(i_{obstacle},\ j_{obstacle}) \leftarrow (i_{intercept},\ j_{intercept})$
24: **break**
25: **end if**
26: **else**
27: **if** ϑ *at* $(i_{intercept},\ j_{intercept})$ *is occupied* **then**
28: $(i_{obstacle},\ j_{obstacle}) \leftarrow (i_{intercept},\ j_{intercept})$
29: **end if**
30: **end if**
31: $error+ = \delta j$
32: **if** $2 * error > \delta i$ **then**
33: $j+ = jstep$
34: $error- = \delta i$
35: **end if**
36: **end for**

37: **return** $z_t^{k*} =$ **grid_resolution** $* \sqrt{(i_{local} - i_{obstacle})^2 + (j_{local} - j_{obstacle})^2}$

Table 4.5: The Bresenham line algorithm.

4.3.3 Resampling

The resampling step has the important function of forcing the particles back to posterior belief distribution $bel(\mathbf{x_t})$. The easiest way to do resampling is to draw with replacement M particles from the temporary set $\overline{\xi}_{\mathbf{t}}$. The probability of drawing each particles is given by its importance weight. This way of doing resampling is analogous to spinning a roulette wheel. Figure 4.2 shows a roulette wheel with M wheel sectors each representing a particle and the width of each wheel sector is sized according to the weight of each particle. During the resampling process, the roulette wheel is spun M times and the particles from the outcomes will be selected into ξ_t.

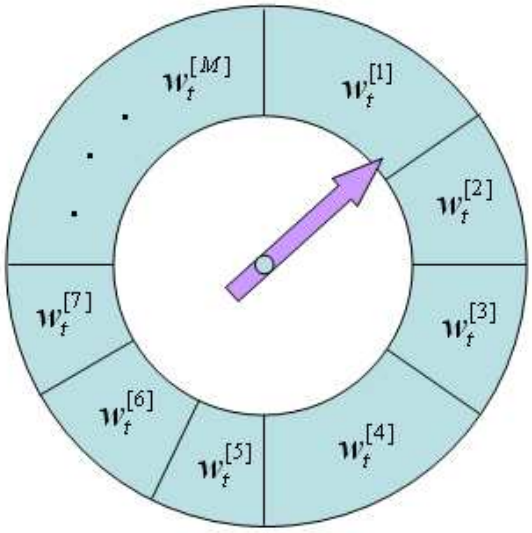


Figure 4.2: Resampling process by drawing the particles with probabilities given by the respective weights.

The beauty of the roulette wheel resampling algorithm lies in the ease of implementation. However, the resampling process tends to induce a loss of diversity in the particle population. A good example is localization of a robot that does not move and with no sensor [10]. Figure 4.3 shows an illustration of this example. Assuming that the true state of the robot is in the center of the environment. The robot is stationary and nine particles with equal initial weights are uniformly distributed in the environment to estimate the pose of the robot. Obviously, the particles will never be able to find out the true pose since the robot possesses no sensors. Therefore, the pose of the particles should remain identical to their initial poses at any point of time. Unfortunately, the resampling step will cause the particles to eventually converge to one pose. The particles are resampled with the same weight from Figures 4.3(a) to 4.3(f). As a result, there will be particles occupying the same pose after the first selection in Figure 4.3(a) and this increases its chance of being selected again. Eventually, all the particles will converge to one pose as shown in Figure 4.3(f). Although the particles may converge to any of the nine initial poses, this does not necessarily means that an accurate estimate of the true pose has been achieved.

The solution to this problem is the *low variance resampling* [10] shown in Table 4.6. The weights of the particles are normalized from Lines 3 to 5. A random number r in the interval $[0;\ M^{-1}]$ is chosen in Line 6. The for loop from Lines 9 to 16 then repeatedly computes U by adding fixed amount of M^{-1} to r and select the i^{th} particle that fulfils

$$\underset{i}{\operatorname{argmin}} \sum_{m=1}^{i} w_t^{[m]} \geq U \tag{4.11}$$

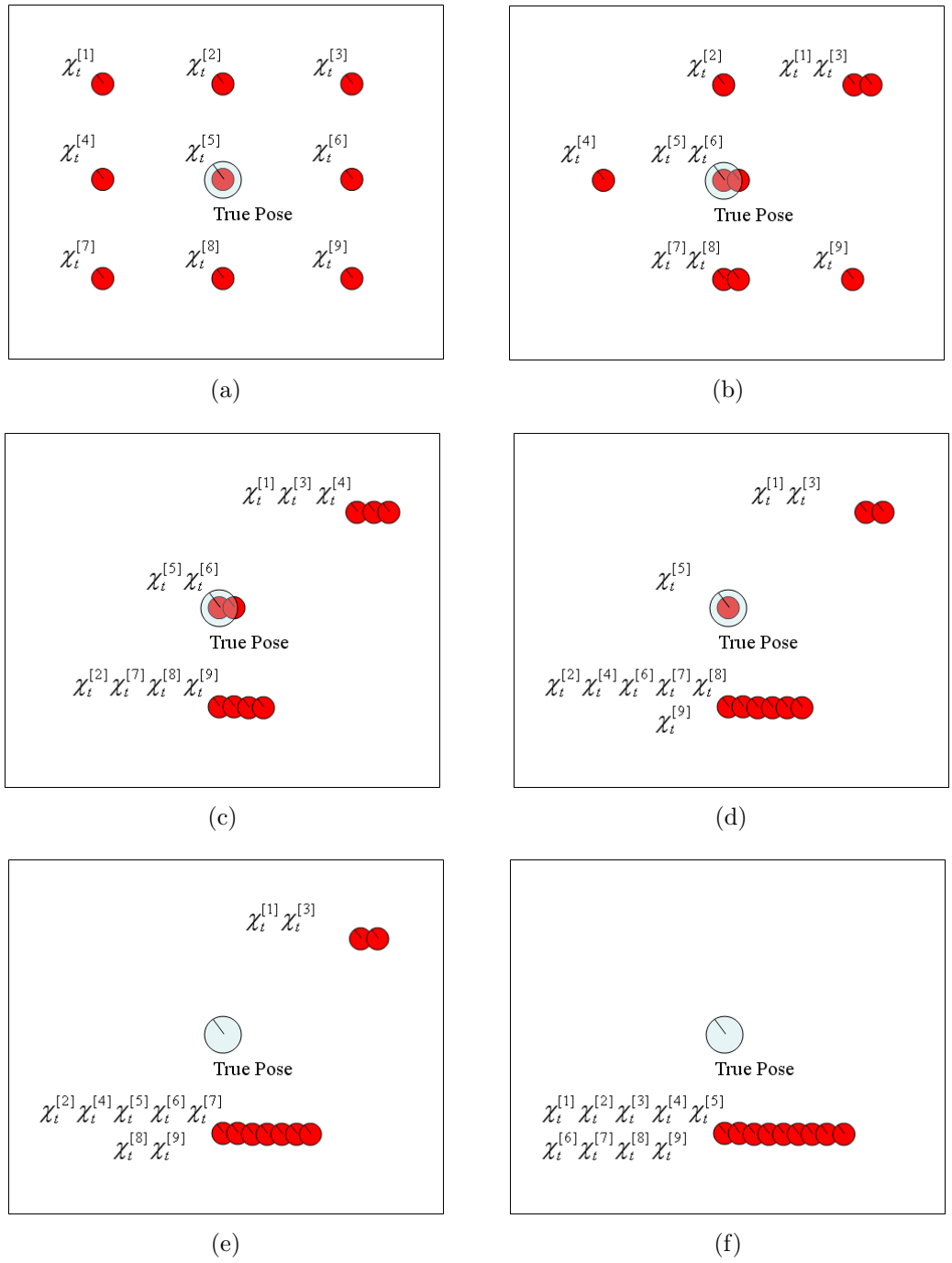


Figure 4.3: The problem of particles convergence due to repetitive resampling despite the robot having no motion and sensors.

The advantage of the low variance resampling algorithm over the roulette wheel resampling algorithm is it covers the sample space in a more systematic fashion. The low variance resampling algorithm selects the particles systematically with a single random number rather than selecting them independently at random. Hence, there will not be any loss of diversity to the particle population. The exact set of particles will remain for the example where the robot has no motion and sensors.

1: **low_variance_resampler**($\overline{\xi}_{\mathbf{t}}$):
2: $\xi_{\mathbf{t}} = \emptyset$
3: **for** $m = 1$ *to* M **do**
4: $\quad w_t^{[m]} = \frac{w_t^{[m]}}{\sum_{i=1}^{M} w_t^{[i]}}$
5: **end for**
6: $r = rand(0, M^{-1})$
7: $c = w_t^{[1]}$
8: $i = 1$
9: **for** $m = 1$ *to* M **do**
10: $\quad U = r + (m-1).M^{-1}$
11: $\quad$ **while** $U > c$ **do**
12: $\quad\quad i = i + 1$
13: $\quad\quad c = c + w_t^{[i]}$
14: $\quad$ **end while**
15: $\quad$ *add* ${\chi_{\mathbf{t}}}^{[i]}$ *to* $\xi_{\mathbf{t}}$
16: **end for**
17: **return** $\xi_{\mathbf{t}}$

Table 4.6: The low variance resampling algorithm.

4.3.4 Pose Estimate

The pose of the robot is estimated from the particle distribution at every time step to complete the localization process. In this thesis, the pose estimate $\mathbf{x_t^{estimate}}$ is

chosen as the weighted mean in a small window around the highest weight particle. This is also known as the robust mean [17]. The highest weight $\mathbf{x_t^{max}}$ particle is first obtained from Equation 4.12.

$$\mathbf{x_t^{max}} = \{\mathbf{x_t^{[m]}} \mid w_t^{[m]} = max(w_t^{[k]}) : k = 1, 2, 3....M\} \tag{4.12}$$

Next, the pose estimate $\mathbf{x_t^{estimate}}$ is computed from all the particles $\mathbf{x_t^{[i]}}$ that are enclosed within a circular window of radius β and center $\mathbf{x_t^{max}}$ as shown in Equation 4.13.

$$\mathbf{x_t^{estimate}} = \{\sum_i \tilde{w}_t^{[i]} \mathbf{x_t^{[i]}} \mid \|\mathbf{x_t^{[i]}} - \mathbf{x_t^{max}}\| \leq \beta\} \tag{4.13}$$

Note that $\tilde{w}_t^{[i]}$ is the weight of the particle $\mathbf{x_t^{[i]}}$, normalized over the total weight of all the particles that are enclosed within the circular window of radius β and center $\mathbf{x_t^{max}}$.

4.4 Simulation and Implementation Results

The simulation and implementation results for the local localization, global localization and kidnapped problem will be shown in Sections 4.4.1 and 4.4.2. The simulations are done in an interactive simulator that is developed by the author. The true pose of the robot is controlled by the user. In addition, the simulator simulates the odometry readings and laser scan measurements. Figures 4.4(a) to 4.4(d) show snapshots of the simulator which represents a 256m x 256m environment in the real world. The white regions represents free spaces and the black regions represent obstacles. The robot starts moving from an initial pose shown in Figure 4.4(a). It can be seen from Figures 4.4(b) to 4.4(d) that the odometry error accumulates as the

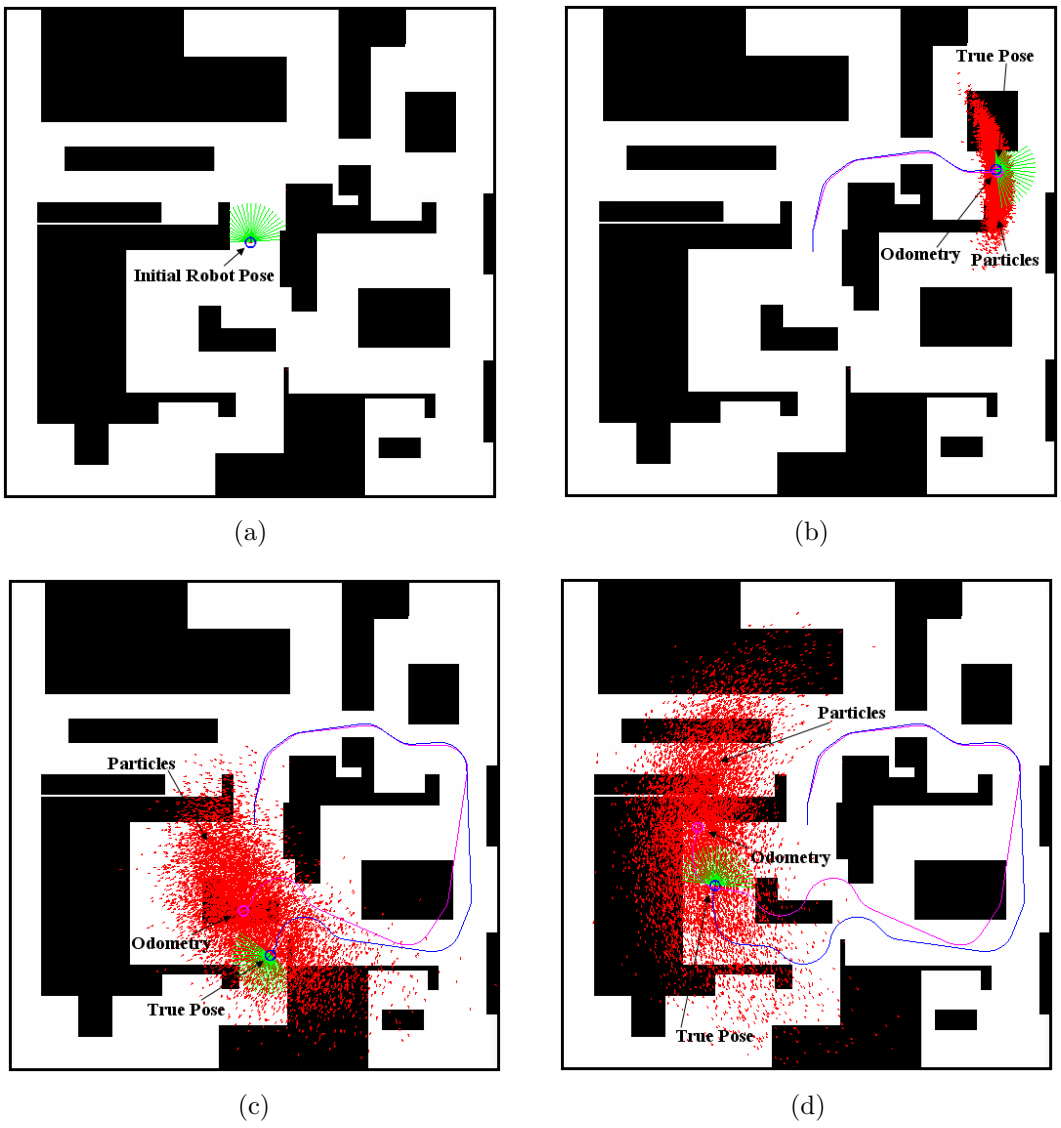
robot moves a longer distance. The sensor measurements are obtained from beams projected from the true pose of the robot. The implementations of the algorithms are done on the ER2 robot equipped with odometer and the URG laser range finder. The motion of the ER2 robot is controlled by the user with a Joystick.

4.4.1 Local Localization

Figures 4.5(a) to 4.5(d) show the simulation results without the localization algorithm. 10000 particles are used to sample the odometry error. The simulation starts with a known initial pose of the robot. Hence, all the particles are initialized to the initial pose as shown in Figure 4.5(a). Notice that the particles set diverges as the robot moves a longer distance from Figures 4.5(b) to 4.5(d). This is due to the accumulated odometry error. The odometry error will eventually grow unbounded as the robot travels greater distances.

Figures 4.6(a) to 4.6(d) show the snapshots of the local localization simulation. The inputs to the particle filter algorithm are the odometry and sensor measurement readings. The particles are initialized to the initial known pose of the robot as shown in Figure 4.6(a). Figures 4.6(b) to 4.6(d) shows that the particles are giving an accurate estimate of the true pose despite the growing odometry error. This is because the predicted measurement $\mathbf{z_t^*}$ from a particle that lies closer to the true pose of the robot has lower discrepancy from the sensor measurement $\mathbf{z_t}$ and hence will be assigned a higher weight (see Section 4.3.2). As a result, the particles that are closer to the true pose of the robot have a higher chance of getting selected during the resampling

Figure 4.4: Snapshots of the interactive virtual simulator that obtains the true pose of the robot from the user as well as simulates the odometry and sensor measurements. Note that the odometry error grows larger as the robot moves a longer distance.



(a) (b)

(c) (d)

Figure 4.5: Snapshots of the odometry error sampled by 10000 particles.

process (see Section 4.3.3).

The implementation of the particle filter to solve the local localization is carried out in the corridor outside the Control and Mechatronics Laboratory 1 on Level 4, EA Block of the NUS Engineering Faculty. The dimension of the corridor is approximately 41.8m x 2.6m. Figure 4.7 shows a picture of the corridor and Figure 4.8 shows the occupancy grid map of the corridor.

Figures 4.9(a) to 4.9(d) show the implementation results of the local localization. The particle set is initialized to the initial known pose of the robot show in Figure 4.9(a). Lesser number of 1000 particles are used in the implementation as compared to the 10000 particles used in the simulation because the implementation environment is much smaller than the simulation environment. Notice that the particles are initialized uniformly within a circle of radius 100mm around the initial position of the robot. The orientation of the particles are also initialized uniformly within $\pm 5^o$ to the initial orientation of the robot. This is to eliminate possible errors in estimating the initial pose of the robot. Figures 4.9(b) to 4.9(d) show that the error from the odometer grows as the robot travels a greater distance. The robot will be thinking that it is traveling in occupied space if it relied solely on the odometry readings and this is obviously wrong. It can also be seen that the particle filter gives a more reasonable pose estimate where the robot is always moving within the free space.

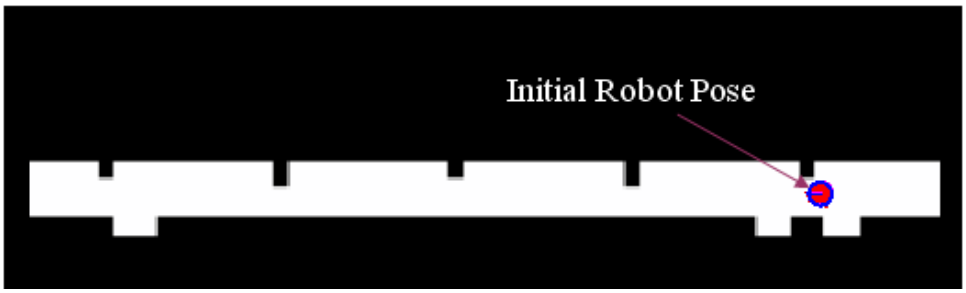
Figure 4.6: Simulation of the local localization problem. The particles are able to give an accurate estimate of the true pose despite the large odometry error.

Figure 4.7: Corridor outside the Control and Mechatronics Laboratory 1.

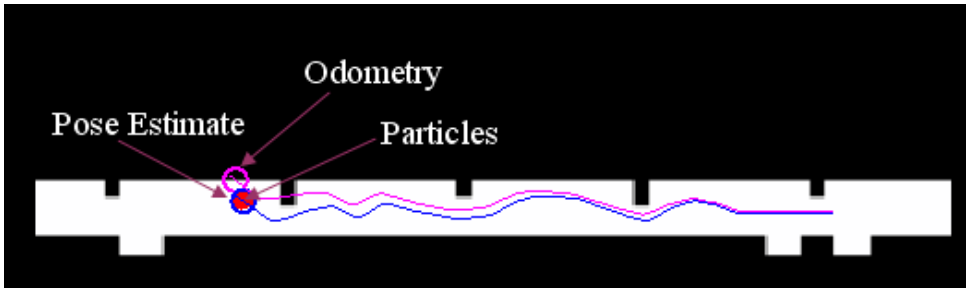
Figure 4.8: Occupancy grid map of the corridor outside the Control and Mechatronics Laboratory 1.

4.4.2 Global Localization and the Kidnapped Problem

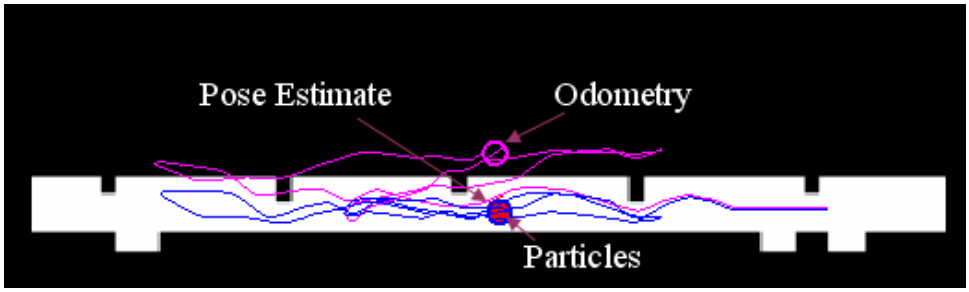
Figures 4.10(a) to 4.10(f) show the simulation results for the global localization problem. The inputs to the global localization algorithm are the odometry and sensor measurement readings. During the start of the operation, the odometry is always



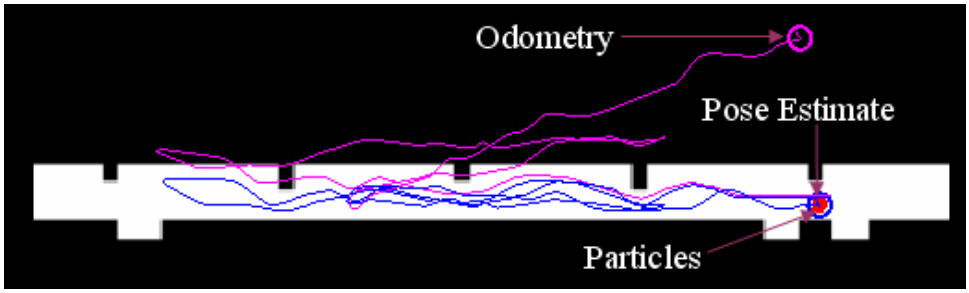
(a)



(b)



(c)



(d)

Figure 4.9: Implementation of the particle filter to solve the local localization problem. 1000 particles are used. The robot starts from its initial pose on the right end of the corridor. It travels to the left end of the corridor, right towards its initial pose, left again and finally travels back to its initial pose. Notice that the error from the odometer grows as the robot travels a greater distance.

reset to $\mathbf{x_t} = [0\ 0\ 0]^T$ and the true pose of the robot could be anywhere within the environment. Hence, the particles are initialized uniformly in the free space of the given map as shown in Figure 4.10(a). The particles eventually converges to the true pose from Figures 4.10(e) and 4.10(f) despite the wrong pose given by the odometry readings. Note that 10000 particles are used in the simulation.

The particle filter algorithm discussed so far is not sufficient for the robot to recover from the kidnapped problem. Fortunately, the problem can be easily solved by observing the total weight of the filter at each iteration. The total weight of the particles are computed before the weights are normalized in the resampling step at each iteration. Figure 4.12 shows the total weight of the particle set for the simulations done in Figures 4.10(a) to 4.10(f) and Figures 4.11(a) to 4.11(f). Note that Figure 4.11 is a continuation of the simulation done in Figure 4.10. The robot is kidnapped in Figure 4.11(a) and this causes a sharp drop in the total weight of the particle set as shown in Figure 4.12. Figure 4.11(b) shows that the particles are re-initialized uniformly in the free space after detecting the kidnapped. The global localization process is repeated from Figures 4.11(c) to 4.11(d) and the particles finally converges at Figures 4.11(e) and 4.11(f).

Figures 4.13(a) to 4.13(h) show snapshots of the implementation results of the global localization and kidnapped problem using 5000 particles along the corridor shown in Figure 4.7. The particles are initialized uniformly in the free space as shown in Figure 4.13(a) and eventually converges from Figures 4.13(b) to 4.13(d). The robot is kidnapped in Figure 4.13(e) and this is reflected in the sharp drop of

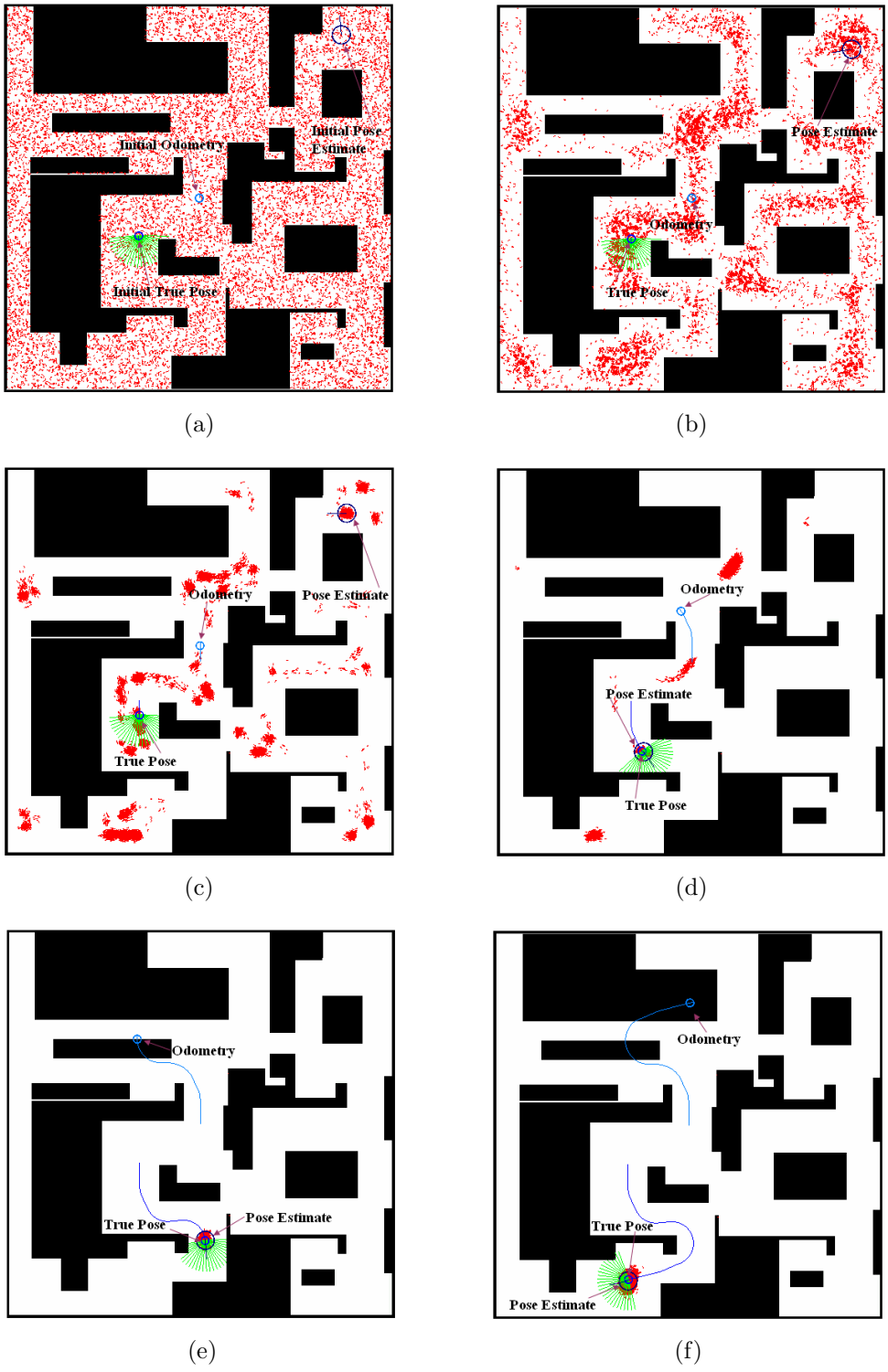


Figure 4.10: Simulation of the global localization problem. The particles are initialized uniformly in the free space because the initial pose of the robot is unknown. The particle set eventually converges to the true pose.

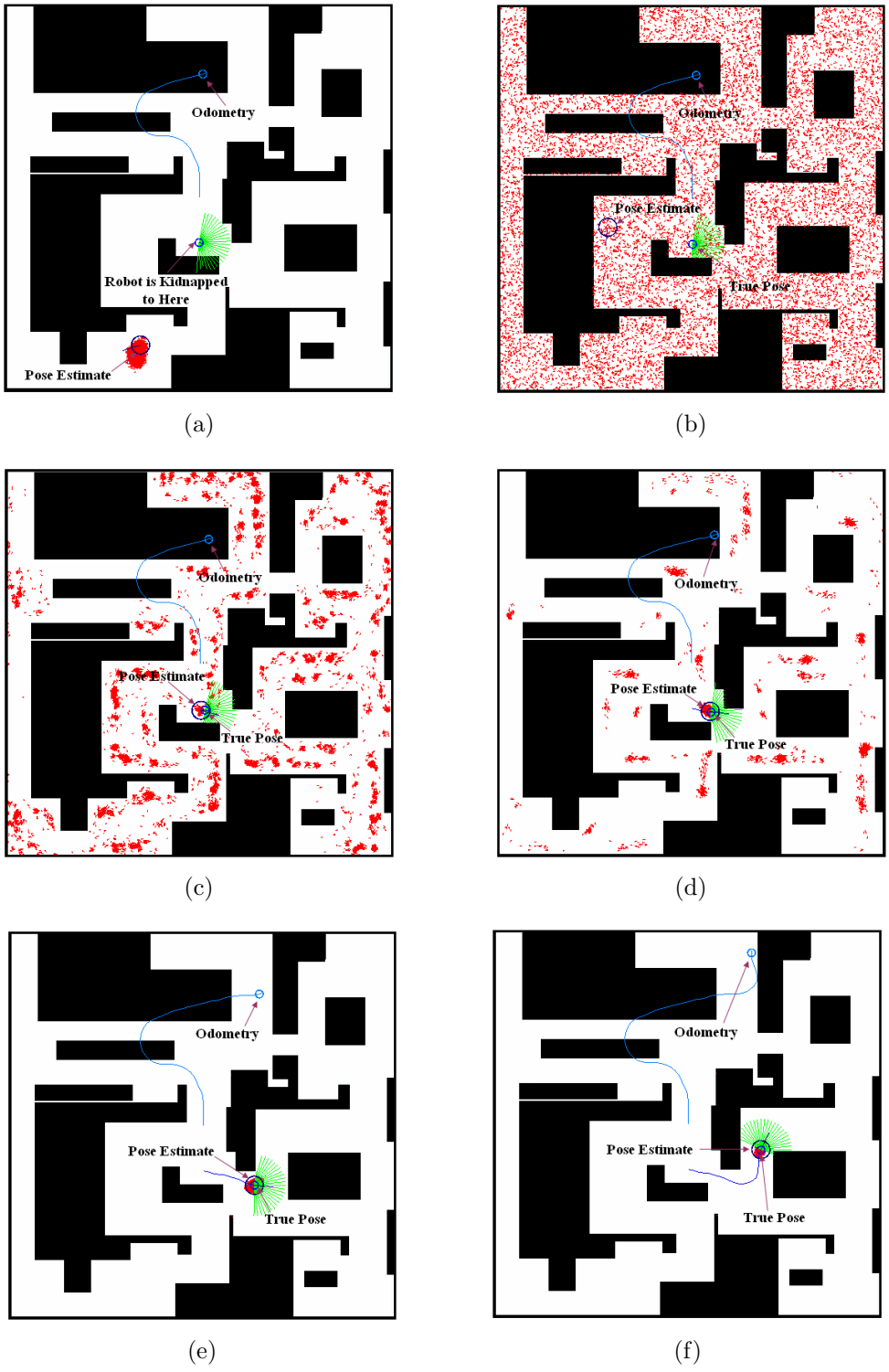


Figure 4.11: The simulation in 4.10 is continued here. The robot is kidnapped to the pose in (a) and causes a sharp drop in the total weight of the particle set. The particles are re-initialized in (b) and finally converges to the true pose in (f).

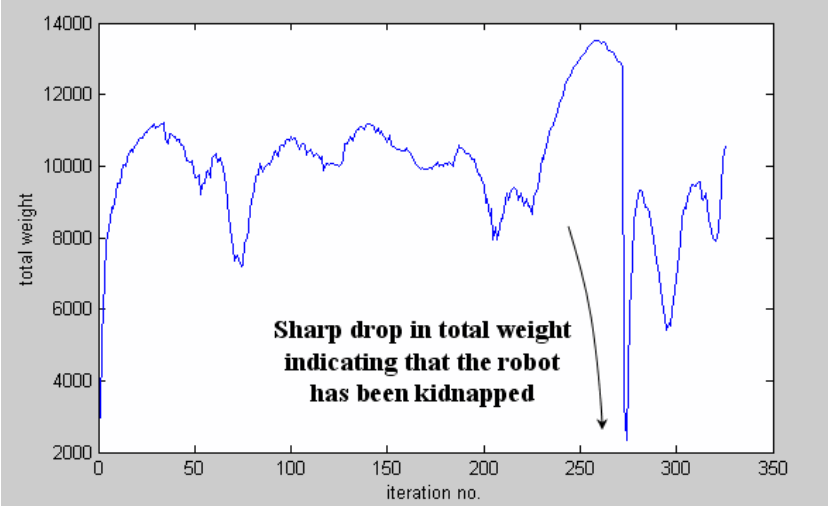


Figure 4.12: Total weights of the particle set recorded over time for the simulation done on the global localization and kidnapped problem shown in Figures 4.10 and 4.11.

the total weights shown in Figure 4.14. The particles are re-initialized uniformly in the free space after the detection of the kidnap. Figures 4.13(f) to 4.13(h) show that the particle set eventually converges to the new robot pose.

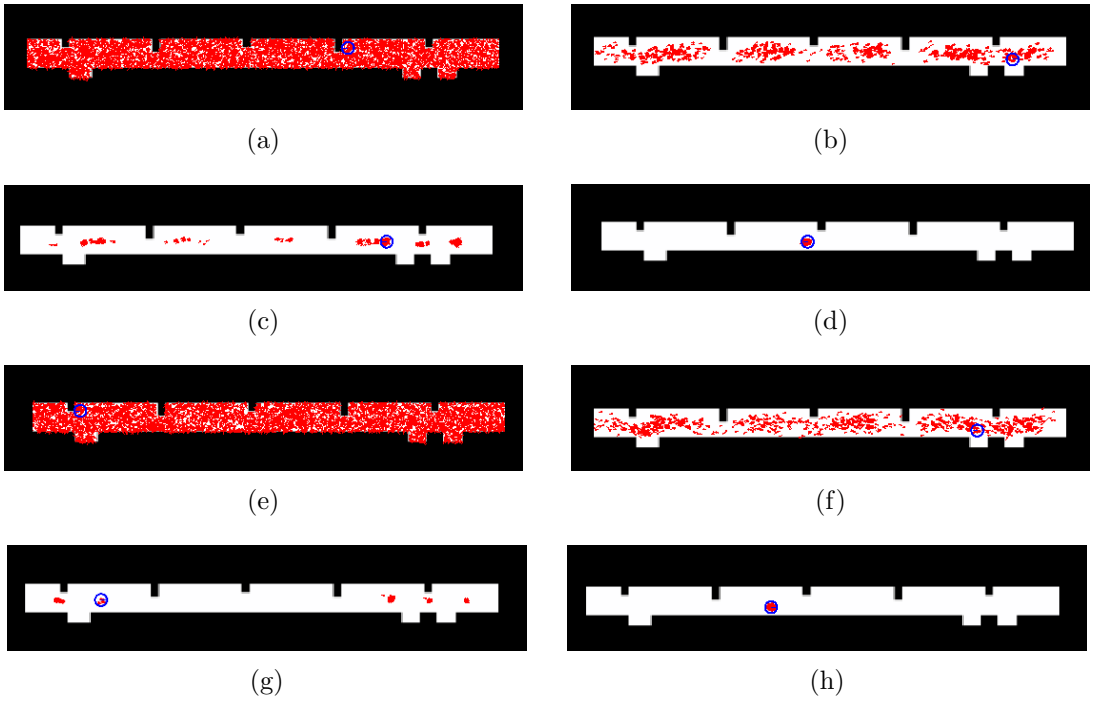


Figure 4.13: Implementation of the global localization and kidnapped problem. The particles are initialized uniformly in the free space to estimate the unknown robot pose in (a). The particles gradually converges from (b) to (d). The robot is kidnapped in (e) and the particles are re-initialized uniformly in the free space. The particles converges to the new pose in (h).

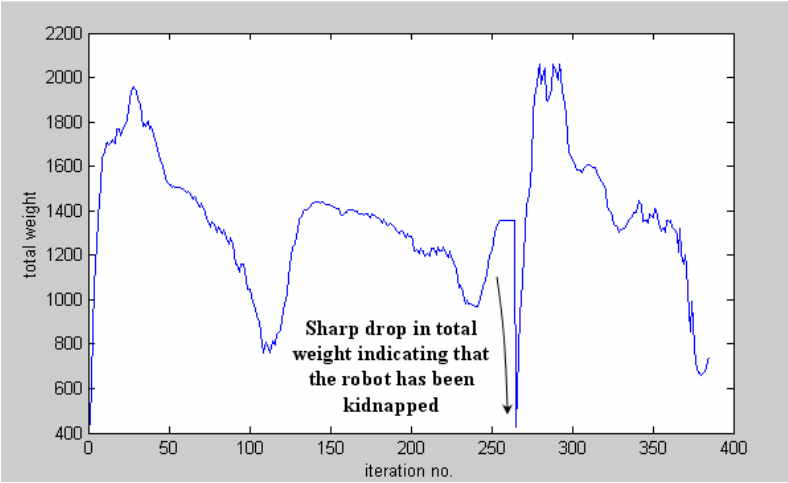


Figure 4.14: Total weights of the particle set recorded over time during implementation of the global localization and kidnapped problem shown in Figure 4.13.

CHAPTER 5

SIMULTANEOUS LOCALIZATION AND MAPPING

5.1 Introduction

The Simultaneous Localization and Mapping (SLAM) problem asks if a mobile robot is able to incrementally build a consistent map ϑ of an unknown environment and simultaneously determines its own pose $\mathbf{x_t}$ within this map. The SLAM problem is significantly more difficult than the local localization, global localization and kidnapped problem that were discussed in Chapter 4. This is because the robot does not have any prior knowledge of the environment and neither does the robot know its initial pose. The robot has to rely on the measurement data $\mathbf{z_{1:t}}$ and control data $\mathbf{u_{1:t}}$ to iteratively build a map of the environment and deduces its pose at each iteration from the available map. Any inaccuracies of the map and pose that are left unchecked will be accumulated, thus grossly distorting the map and therefore ruining the robot's ability to deduce its pose in further iterations.

In the probabilistic form, the SLAM problem requires the probability distribution

$$p(\mathbf{x_t}, \vartheta \mid \mathbf{z_{1:t}}, \mathbf{u_{1:t}}) \tag{5.1}$$

to be computed at all times t. This probability distribution describes the joint probability distribution of the robot state $\mathbf{x_t}$ and the map ϑ of the environment given all the measurement data $\mathbf{z_{1:t}}$ and control data $\mathbf{u_{1:t}}$. In general, a recursive solution to the SLAM problem is desired [18].

Another problem associated with SLAM is the loop closure problem [15, 19, 20] in large cyclic environment. The loop closure problem arises due to the accumulation of errors during the SLAM process. As a result, a robot traveling through an unknown terrain may not be able to decide whether or not it has returned to a previously visited location. The solution to the loop closure problem seeks to provide a mean for the robot to decide whether or not it has returned to an area that was previously visited based on its current measurement data $\mathbf{z_t}$, state $\mathbf{x_t}$ and the acquired map ϑ. In addition, the robot must be able to correct the acquired map ϑ upon detection that it has returned to a previously visited location.

In this chapter, two of the existing SLAM algorithms - the *extended kalman filter* (EKF) and the *FastSLAM* algorithm will be discussed in Section 5.2. In Section 5.3, the details of building an occupancy grid map ϑ of the environment from the measurement data $\mathbf{z_{1:t}}$ will be described. The occupancy grid mapping algorithm assumes that the precise pose of the robot is known at all times. Details and implementation results of a novel SLAM algorithm will be given in Section 5.4.

5.2 Related Works

Two of the popular SLAM algorithms - SLAM with the EKF and FastSLAM algorithm will be discussed in this section.

5.2.1 SLAM with Extended Kalman Filter

The earliest and perhaps the most influential SLAM algorithm is based on the EKF [10]. The idea of using the EKF to solve the SLAM problem was first proposed by Cheeseman and Smith in 1986 [21] and first implemented by Leonard and Whyte in 1991 [1].

The EKF SLAM algorithm is similar to the EKF localization algorithm (see Section 4.2.1). Both algorithms use feature-based maps [36, 37, 38] and assume that the state belief follows a Gaussian distribution represented by the mean and covariance. The main difference between the two algorithms is that in addition to estimating the robot pose $\mathbf{x_t}$, the EKF algorithm also estimates the coordinates of all features encountered along the way. This is done by including the coordinates of the features $\vartheta = [\vartheta_{1,x}\ \vartheta_{1,y}\ \vartheta_{2,x}\ \vartheta_{2,y}\ ...\ \vartheta_{N,x}\ \vartheta_{N,y}]^T$ into the state vector. Note that N is the number of acquired map coordinates at current time t. The resulting state vector shall be known as the combined state vector and denoted by

$$\mathbf{y_t} = [\mathbf{x_t}\ \vartheta]^T = [x\ y\ \theta\ \ \vartheta_{1,x}\ \vartheta_{1,y}\ \vartheta_{2,x}\ \vartheta_{2,y}\ ...\ \vartheta_{N,x}\ \vartheta_{N,y}]^T \tag{5.2}$$

The combined state belief $bel(\mathbf{y_t})$ is computed recursively from the prediction and update step. In the prediction step, the predicted belief distribution $\overline{bel}(\mathbf{y_t})$ represented by the predicted mean $\overline{\mathbf{y}}_\mathbf{t}$ and covariance $\overline{\mathbf{S}}_\mathbf{t}$ is computed from

$$\overline{\mathbf{y}}_\mathbf{t} = \mathbf{f}(\mathbf{y_{t-1}},\ \mathbf{u_t}) \tag{5.3}$$

$$\overline{\mathbf{S}}_\mathbf{t} = \mathbf{F_t}\ \mathbf{S_{t-1}}\ \mathbf{F_t}^T + \mathbf{Q_t} \tag{5.4}$$

where $\mathbf{f}(.)$ is the motion model of the robot and $\mathbf{F_t}$ is the Jacobian of $\mathbf{f}(.)$ evaluated at $\mathbf{y_{t-t}}$ and $\mathbf{Q_t}$ is the covariance of the motion model.

The EKF SLAM algorithm will check for any newly acquired features and incorporate them into the predicted belief prior to the update step. The predicted belief $\overline{bel}(\mathbf{y_t})$ is subsequently transformed into the desired belief $bel(\mathbf{y_t})$ by incorporating the sensor measurement $\mathbf{z_t}$ in the update step given by

$$\mathbf{K_t} = \overline{\mathbf{S}}_\mathbf{t}\mathbf{H_t}^T(\mathbf{H_t}\overline{\mathbf{S}}_\mathbf{t}\mathbf{H_t}^T + \mathbf{R_t})^{-1} \tag{5.5}$$

$$\mathbf{y_t} = \overline{\mathbf{y}}_\mathbf{t} + \mathbf{K_t}(\mathbf{z_t} - \mathbf{h}(\overline{\mathbf{y}}_\mathbf{t}, \vartheta)) \tag{5.6}$$

$$\mathbf{S_t} = (\mathbf{I} - \mathbf{K_t}\mathbf{H_t})\overline{\mathbf{S}}_\mathbf{t} \tag{5.7}$$

$\mathbf{K_t}$ is the Kalman gain for the EKF SLAM algorithm. $\mathbf{h}(.)$ is the sensor measurement model, and $\mathbf{H_t}$ is the Jacobian of $\mathbf{h}(.)$ evaluated at $\mathbf{y_{t-1}}$. $\mathbf{R_t}$ is covariance of the sensor measurement model.

5.2.2 FastSLAM

The FastSLAM algorithm that was introduced by Montemerlo [22] marked a fundamental conceptual shift in the design of recursive probabilistic SLAM [18]. Previous efforts in SLAM algorithms focus on improving the performance of the EKF SLAM while retaining its linear Gaussian assumptions. The FastSLAM algorithm was the first to use the *Rao-Blackwellized* particle filter to represent the state belief $bel(\mathbf{x_t})$ with a non-Gaussian distribution along with Gaussians to represent map features.

Figure 5.1 shows the denotation of the Rao-Blackwellized particles in the FastSLAM algorithm. Each particle in FastSLAM contains an estimated robot pose, denoted by $\mathbf{x_t^{[m]}}$, and a set of EKF with a pair of mean $\mu_{\mathbf{j,t}}^{\mathbf{[m]}}$ and covariance $\mathbf{\Sigma_{j,t}^{[m]}}$ representing the location of the j^{th} features ϑ_j of the map ϑ.

	Pose	**Feature 1**	**Feature 2**	...	**Feature N**
Particle $m = 1$	$x_t^{[1]} = \left(\begin{bmatrix} x & y & \theta \end{bmatrix}^T\right)_t^{[1]}$	$\mu_1^{[1]}, \Sigma_1^{[1]}$	$\mu_2^{[1]}, \Sigma_2^{[1]}$	...	$\mu_N^{[1]}, \Sigma_N^{[1]}$
Particle $m = 2$	$x_t^{[1]} = \left(\begin{bmatrix} x & y & \theta \end{bmatrix}^T\right)_t^{[2]}$	$\mu_1^{[2]}, \Sigma_1^{[2]}$	$\mu_2^{[2]}, \Sigma_2^{[2]}$	...	$\mu_N^{[2]}, \Sigma_N^{[2]}$
		.			
		.			
		.			
Particle $m = M$	$x_t^{[1]} = \left(\begin{bmatrix} x & y & \theta \end{bmatrix}^T\right)_t^{[M]}$	$\mu_1^{[M]}, \Sigma_1^{[M]}$	$\mu_2^{[M]}, \Sigma_2^{[M]}$	...	$\mu_N^{[M]}, \Sigma_N^{[M]}$

Figure 5.1: Rao-Blackwellized particles in FastSLAM, M denotes the total number of particles.

Similar to the particle filter algorithm for localization, the FastSLAM algorithm computes the state belief $bel(\mathbf{x_t})$ recursively with the prediction and update steps. In the prediction step, the hypothetical state $\mathbf{x_t^{[m]}}$ of the m^{th} particle is generated by sampling from the odometry motion model (see Section 4.3.1). The set of particles obtained after M iterations is the discrete representation of the predicted belief $\overline{bel}(\mathbf{x_t})$.

In the update step, the mean $\mu_{\mathbf{j,t}}^{\mathbf{[m]}}$ and covariance $\mathbf{\Sigma_{j,t}^{[m]}}$ of the observed features are updated with the sensor $\mathbf{z_t}$ measurement and predicted measurement $\mathbf{z_t^*}$ using the standard EKF update equations (see Section 4.2.1). The importance factor $w^{[m]}$ of the m^{th} particle is given by a Guassian over the sensor measurement $\mathbf{z_t}$ with the predicted measurement $\mathbf{z_t^*}$ and measurement covariance $\mathbf{Q_t}$ as the mean and covariance.

$$w^{[m]} = det(2\pi\mathbf{Q_t})^{-\frac{1}{2}}\exp\{-\frac{1}{2}(\mathbf{z_t} - \mathbf{z_t^*})^T\mathbf{Q_t^{-1}}(\mathbf{z_t} - \mathbf{z_t^*})\} \tag{5.8}$$

Finally, the particles are transformed into the belief distribution $bel(\mathbf{x_t})$ after the resampling step (see Section 4.3.3).

5.3 Occupancy Grid Mapping

A solution to the SLAM problem is necessary in the absence of both an initial map and exact pose information. The robot has to estimate the map and localize itself relative to this map. Solutions to the SLAM problem therefore have to be built on top of two problems decoupled from SLAM. First, the localization problem with a known map of the environment and second, the map building problem with known

pose of the robot. Solutions to the localization problem with a known map are discussed in the previous chapter. In this section, the map building problem with the assumption that the exact pose of the robot is know will be discussed.

Map building is the process of generating consistent maps from noisy and uncertain measurement data when the pose of the robot is known. The occupancy-grid mapping algorithm [10, 41, 42, 43, 44] is a popular choice for map building because it is comprehensive and easy to implement. Occupancy-grid mapping represents the environment as a tessellation of rectangloid grid cells where each cell corresponds to an area in the physical environment. Let ϑ denotes the occupancy grid map and ϑ_i denotes the grid cell with index i.

The objective of the occupancy grid mapping is to estimate the occupancy value of each grid cell denoted by $p(\vartheta_i \mid \mathbf{z_{1:t}}, \mathbf{x_{1:t}})$, where $\mathbf{z_{1:t}}$ is the set of sensor measurement data up to time t and $\mathbf{x_{1:t}}$ is the sequence of all poses of the robot. Occupancy values indicate the probability of whether the cell is occupied $p(\vartheta_i \mid \mathbf{z_{1:t}}, \mathbf{x_{1:t}}) = 1$ or free $p(\vartheta_i \mid \mathbf{z_{1:t}}, \mathbf{x_{1:t}}) = 0$. An occupancy value of $p(\vartheta_i \mid \mathbf{z_{1:t}}, \mathbf{x_{1:t}}) = 0.5$ indicates that the cell is an unexplored area. The estimation of the occupancy value for each grid cell is assumed to be independent of other grid cells.

A robot does not have any knowledge of the world when it was first placed in an unknown environment. It is therefore intuitive to set $p(\vartheta_i \mid \mathbf{z_{1:t}}, \mathbf{x_{1:t}}) = 0.5$ for all grid cells at time $t = 0$. The map is updated via the log odds [10, 41, 42, 43, 44] representation of occupancy. The advantage of log odds representation is that it

can avoid numerical instabilities for probability near 0 or 1. The i^{th} grid cell that intercepts the line-of-sight of the sensor measurement is updated according to

$$l_{t,i} = l_{t-1,i} + l_{sensor} \tag{5.9}$$

where $l_{t-1,i}$ is the log odds computed from the occupancy value of the cell at $t-1$.

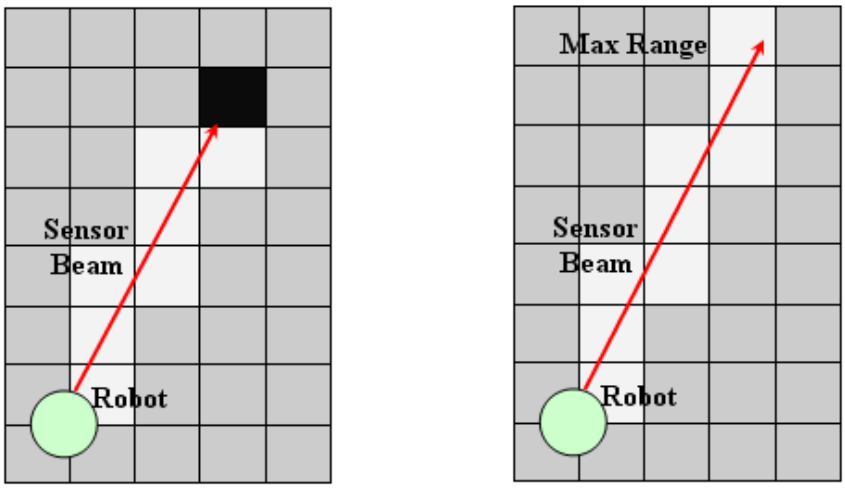
$$l_{t-1,i} = log \frac{p(\vartheta_i \mid \mathbf{z_{1:t-1}}, \mathbf{x_{1:t-1}})}{1 - p(\vartheta_i \mid \mathbf{z_{1:t-1}}, \mathbf{x_{1:t-1}})} \tag{5.10}$$

The value of l_{sensor} depends on the sensor measurement. If the sensor measurement is lesser than the maximum range of the sensor, it means that an object has been detected. In this case, $l_{sensor} = l_{occ}$ for the cell that corresponds to the sensor measurement and $l_{sensor} = l_{free}$ for all the other cells that intercepts the line-of-sight of the sensor measurement.

Figure 5.2(a) shows an illustration of a sensor measurement that is lesser than the maximum range of the sensor. The cell that corresponds to the sensor measurement has been assigned $l_{sensor} = l_{occ}$ and shaded black. All the other cells that intercept the line-of-sight of the sensor measurements are assigned $l_{sensor} = l_{free}$ and shaded white.

If the sensor measurement is equal to the maximum range of the sensor, it means that no object has been detected. In this case, $l_{sensor} = l_{free}$ for all the cells that intercepts the line-of-sight of the sensor measurement. Figure 5.2(b) shows an illustration of a sensor measurement that is equal to the maximum range of the sensor. All the cells that intercepts the line-of-sight of the sensor measurement are assigned

$l_{sensor} = l_{free}$ and shaded white.



(a) (b)

Figure 5.2: Updating an occupancy grid map (a) when an obstacle is detected (b) when a maximum range measurement is detected, i.e. it is assumed that in this case no obstacle is detected.

l_{occ} and l_{free} are computed from

$$l_{occ} = log\frac{p(\vartheta_i = 1)}{1 - p(\vartheta_i = 1)} \tag{5.11}$$

$$l_{free} = log\frac{p(\vartheta_i = 0)}{1 - p(\vartheta_i = 0)} \tag{5.12}$$

where $p(\vartheta_i = 1)$ and $p(\vartheta_i = 0)$ denote the probabilities of the sensor measurement correctly deducing whether a grid cell is occupied or empty. The two probabilities must add up to 1 and their values depend on the accuracy of the sensor. $p(\vartheta_i = 1)$ and $p(\vartheta_i = 0)$ will have values closer to 1 and 0 for an accurate sensor. The values

of $p(\vartheta_i = 1)$ and $p(\vartheta_i = 0)$ remain constant in the map building process. $p(\vartheta_i = 1)$ is assigned a value of 0.9 and $p(\vartheta_i = 0)$ assigned a value of 0.1 for the URG laser range finder used in this research since it is an accurate sensor. The occupancy value of a grid cell is easily recovered from

$$p(\vartheta_i \mid \mathbf{z_{1:t-1}}, \mathbf{x_{1:t-1}}) = 1 - \frac{1}{1 + exp\{l_{t,i}\}} \tag{5.13}$$

5.4 A Novel SLAM Algorithm

Many existing SLAM algorithms such as EKF and FastSLAM are feature-based SLAM and the success of these algorithms depend greatly on feature extractions from raw sensor measurements data. In this section, a SLAM algorithm that overcomes the restrictions from feature extractions by using occupancy grid map will be discussed. This SLAM algorithm has two novel aspects. First, a novel laser scan matching algorithm is introduced to estimate the probability distribution $p(\mathbf{x_t}, \vartheta \mid \mathbf{z_{1:t}}, \mathbf{u_{1:t}})$ recursively. Second, a novel loop closure detection algorithm is used to detect loop closure opportunity and a loop closure algorithm is used to close any detected loops in the map. Detailed descriptions of the scan matching, loop closure detection and loop closure algorithms are found in Sections 5.4.1, 5.4.2 and 5.4.3 respectively.

5.4.1 Laser Scan Matching with Particle Filter

Figure 5.3 shows an occupancy map of a cyclic environment (Level 3 EA Block of the NUS Engineering Faculty) built from raw odometry readings. The robot has returned to previously visited locations and therefore should close the loop in the

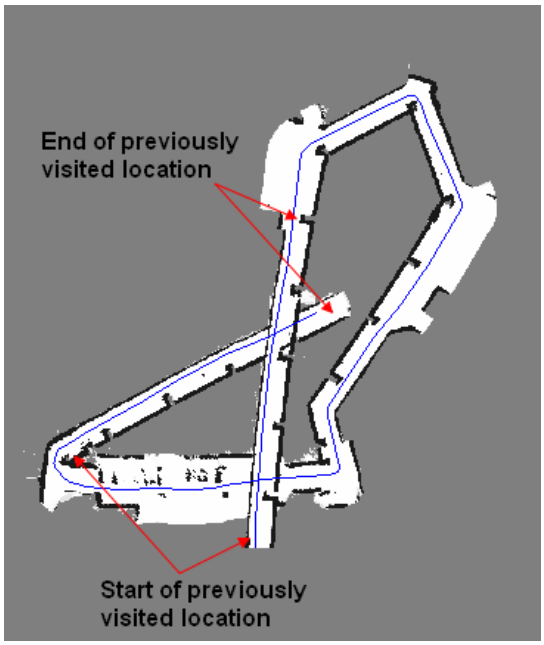


Figure 5.3: An occupancy grid map of Level 3 EA Block of the NUS Engineering Faculty built with raw odometry readings. The robot has returned to previously visited areas and these areas should coincide.

map. However, the accumulation of the odometry errors has grossly distorted the map thus making it hard for the robot to detect the loop closure opportunities. The loop closure detection problem can be made easier by doing laser scan matching. In laser scan matching, the pose of the robot that captures the current laser scan is sought with respect to a reference scan by adjusting the pose of the robot until the best overlap with a reference scan is achieved. Consequently, the short term odometry errors that causes the misalignment between the current and reference scans are reduced. This reduction of the odometry errors by scan matching results in a smaller loop closure error hence reducing the difficulty for loop closure detection (see Section 5.4.2).

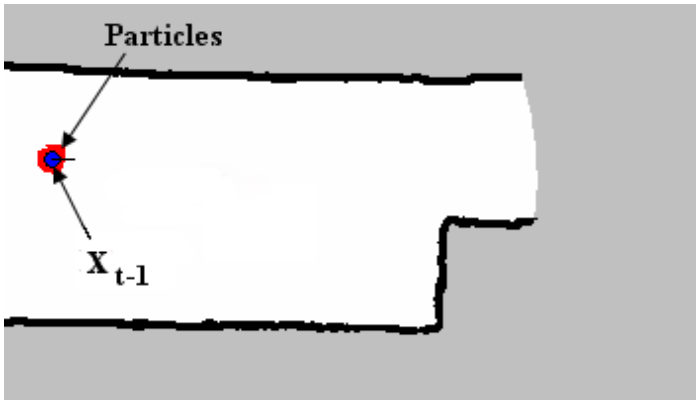
Many scan matching algorithms [45, 46, 47, 48] have been proposed by different researchers over the years. For example, the *iterative closest point* (ICP) algorithm [45], *iterative matching range point* (IMRP) algorithm [46] and *iterative dual correspondence* (IDC) algorithm [46]. In ICP, each point from the current scan is first matched with their respective closest point from the reference scan. Next, an error function computed based on the Euclidean distances between each pair of match points. This error function measures the discrepancy between the scans. Finally, the process is iterated until a match with the least error has been found. The IMRP algorithm is similar to the ICP algorithm except that each point from the current scan is matched with a point from the reference scan that is within a matching range. The IDP algorithm proposed a combination of the ICP and IMRP algorithms by using the ICP to calculate translation and IMRP to calculate rotation. Many of these existing scan matching algorithms rely on intensive iterations and may not be suitable for real time operations.

A novel scan matching algorithm has been proposed and implemented by the author. This scan matching algorithm uses a particle filter which is similar to the one used for localization of a mobile robot in a known environment (see Chapter 4). The algorithm seeks to generate the posterior distribution $p(\mathbf{x_t}, \vartheta \mid \mathbf{z_{1:t}}, \mathbf{u_{1:t}})$ that represents the robot state that yields the best overlap between the current and reference scans. Note that reference scan refers to the acquired map ϑ_{t-1} at time $t-1$. The posterior distribution $p(\mathbf{x_t}, \vartheta \mid \mathbf{z_{1:t}}, \mathbf{u_{1:t}})$ is represented by a finite set of M weighted particles denoted by $\xi_t := \chi_t^{[1]}, \chi_t^{[2]}, ..., \chi_t^{[M]}$ drawn from this posterior. $\chi_t^{[m]} = [\mathbf{x_t^{[m]}}\ w^{[m]}]^T$ denotes the m^{th} particle where $\mathbf{x_t^{[m]}}$ and $w^{[m]}$ are the state and

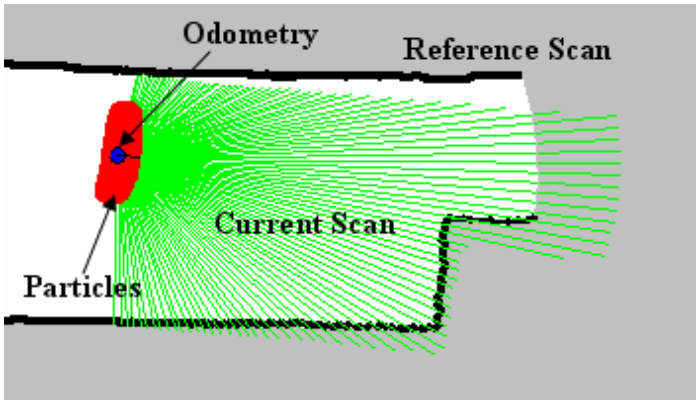
importance factor of this particle.

Particles representing the posterior distribution $p(\mathbf{x_t}, \vartheta \mid \mathbf{z_{1:t}}, \mathbf{u_{1:t}})$ are generated in three steps. First, a temporary particle set $\overline{\xi}_t$ which represents the predicted belief is generated from the odometry motion model $p(\mathbf{x_t} \mid \mathbf{u_t},\ \mathbf{x_{t-1}^{[m]}})$ of the robot. The hypothetical states of the particles can be seen as the search space for the robot state $\mathbf{x_t}$ that yields the best overlap between the current and reference scans. Detailed descriptions of the odometry model can be found in Section 4.3.1. It is important to note that σ_1, σ_2, σ_3 and σ_4 which denote standard deviation of the odometry motion model during pure rotation, pure translation as well as when there are both translation and rotation have the same values as the standard deviations discussed in Section 4.3.1. This is because the same ER2 robot is used to implement the scan matching with particle filter algorithm. Figures 5.4(a) and 5.4(b) show the particle distribution before and after sampling from the odometry motion model. Figure 5.4(b) shows the distribution of the particles which represents the search space for the robot state $\mathbf{x_t}$ that yields the best overlap between the current and reference scans. Note the discrepancy between the current and reference scans.

Second, the importance factor of each particle is computed from the sensor measurement model $p(\mathbf{z_t} \mid \mathbf{x_t^{[m]}},\ \vartheta_{t-1})$. The sensor measurement model is a measure of the similarity between the current scan and reference scan as seen from the hypothetical state of the particle. The higher the similarity between the scans, the higher the weight value. The sensor measurement model used for scan matching is different from the one used for localization of a mobile robot in a known environment $p(\mathbf{z_t} \mid \mathbf{x_t^{[m]}},\ \vartheta)$



(a)



(b)

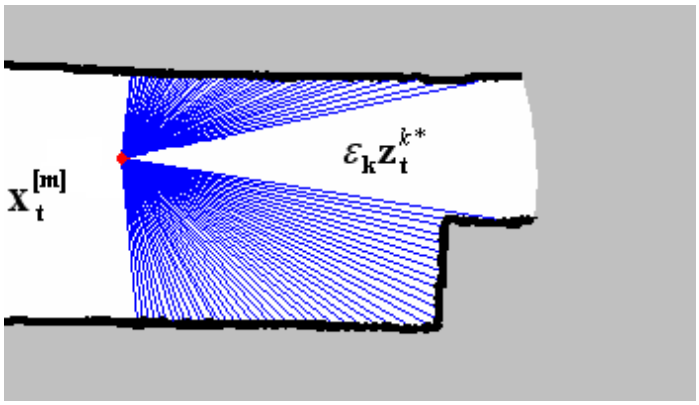
Figure 5.4: Particles distribution before (a) and after (b) sampling from the odometry motion model. The distribution of the particles in (b) represents the search space for the robot state $\mathbf{x_t}$ that yields the best overlap between the current and reference scans. Note also the discrepancy between the current and reference scans.

(see Section 4.3.2). The sensor measurement model for the scan matching algorithm computes the probability conditioned on the map ϑ_{t-1} acquired at the previous time step $t-1$ instead of a known map ϑ. The sensor measurement model $p(\mathbf{z_t} \mid \mathbf{x_t^{[m]}}, \vartheta_{t-1})$ for the scan matching process is given by

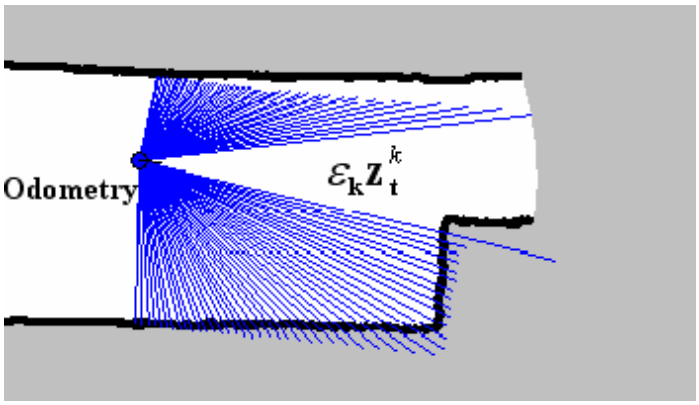
$$p(z_t^k \mid \mathbf{x_t^{[m]}},\ \vartheta_{t-1}) = \frac{\varepsilon_k}{\sqrt{2\pi\sigma_{hit}^2}} e^{-\frac{1}{2}\frac{(z_t^k - z_t^{k*})^2}{\sigma_{hit}^2}} + (1 - \varepsilon_k) \quad (5.14)$$

$$p(\mathbf{z_t} \mid \mathbf{x_t^{[m]}},\ \vartheta_{t-1}) = \eta \prod_{k=1}^{K} p(z_t^k \mid \mathbf{x_t^{[m]}},\ \vartheta_{t-1}), \quad \text{where K} = 513 \quad (5.15)$$

σ_{hit} is the standard deviation for the sensor measurement model and was determined experimentally to be 50mm for the URG laser sensor used in this research (see Section 4.3.2). z_t^k is the k^{th} measurement reading from the laser sensor measurements $\mathbf{z_t} = [z_t^1\ z_t^2\ ...\ z_t^K]^T$ for $K = 513$ and z_t^{k*} is the k^{th} measurement reading from the predicted measurements $\mathbf{z_t^*} = [z_t^{1*}\ z_t^{2*}\ ...\ z_t^{K*}]^T$ for $K = 513$. The predicted measurements are obtained by the ray casting process (see Section 4.3.2) from the hypothetical state $\mathbf{x_t^{[m]}}$ of the robot and the map ϑ_{t-1} acquired at time $t-1$. Note that η in Equation 5.15 is a normalizer that ensures the probability stays within 0 to 1. The most important variable in Equations 5.14 and 5.15 is perhaps ε_k. It is a binary operator which is equals to 1 if both z_t^k and z_t^{k*} are lesser than the maximum range of the laser sensor and 0 otherwise. This is to ensure that the comparison of the current and reference scans is made at the segments where both scans shows signs of existence of obstacles. The existence of obstacles is deduced from the sensor measurements which are lesser than the maximum range of the laser sensor. $\varepsilon_k = 0$ if both z_t^{k*} and z_t^k are not lesser than the maximum laser range. As a result, $p(z_t^k \mid \mathbf{x_t^{[m]}},\ \vartheta_{t-1}) = 1$ and this will not cause any changes to the final value of $p(\mathbf{z_t} \mid \mathbf{x_t^{[m]}},\ \vartheta_{t-1})$. Figure 5.5 shows an example for the selection of the relevant measurements. The measurements where both z_t^{k*} and z_t^k are not lesser than the maximum laser sensor range are omitted.



(a)



(b)

Figure 5.5: (a) Predicted measurement $\mathbf{z_t^*}$ obtained by rays casted from a hypothetical state $\mathbf{x_t^{[m]}}$. (b) Laser sensor measurement $\mathbf{z_t}$ from the odometry reading. Notice that the measurements where both z_t^{k*} and z_t^k are not lesser than the maximum laser sensor range are omitted.

Third, the temporary particle set $\overline{\xi}_t$ undergoes a resampling process where it is transformed into the particle set ξ_t which represents the posterior distribution $p(\mathbf{x_t}, \vartheta \mid \mathbf{z_{1:t}}, \mathbf{u_{1:t}})$. The low variance resampling algorithm described in Section 4.3.3 is used. Finally, the robot state $\mathbf{x_t}$ is chosen as the weighted mean in a small window around the highest weight particle. This is the robust mean and the detailed descriptions can be found in Section 4.3.4. Figure 5.6 shows the final map after the scan matching process where the current range scan has been integrated into the map at the $\mathbf{x_t}$.

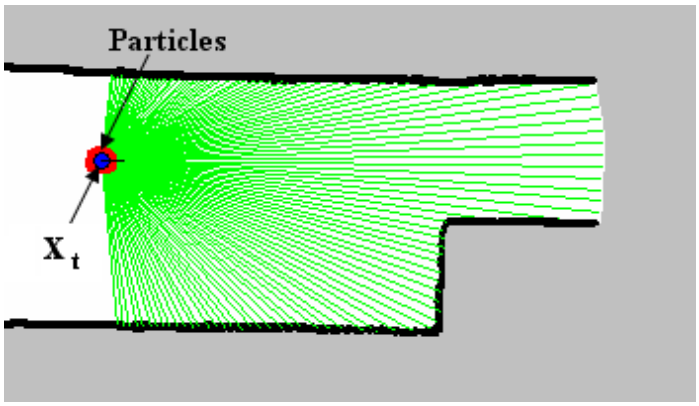
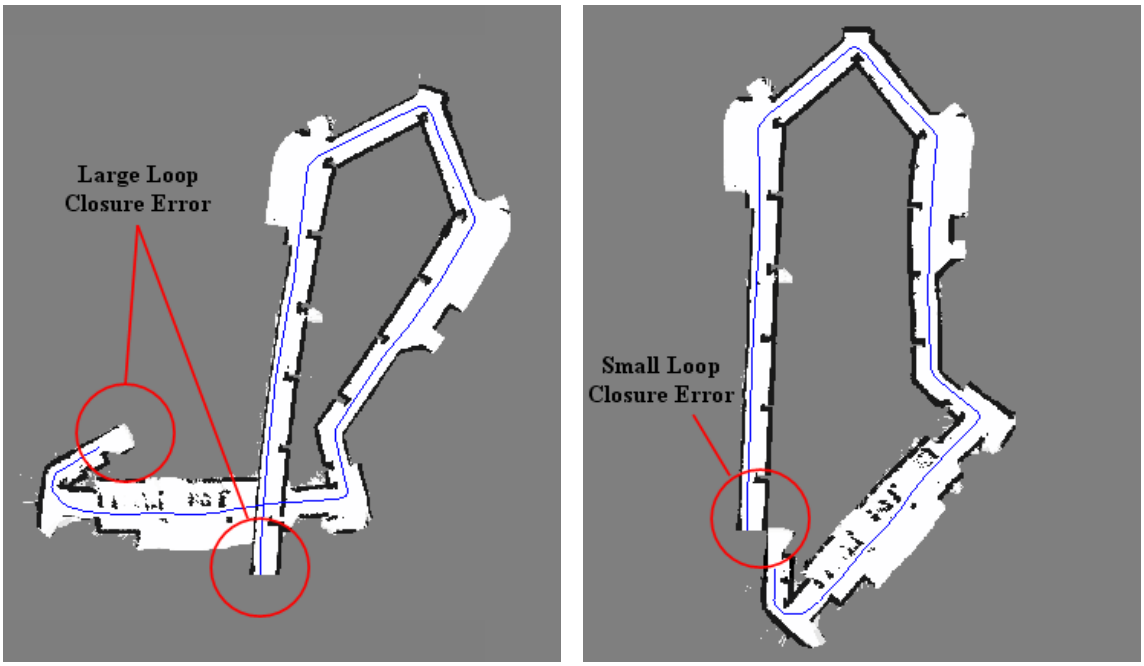


Figure 5.6: Map ϑ_t after the scan matching process where the current range scan has been integrated into the map at $\mathbf{x_t}$. Note that $\mathbf{x_t}$ is the robot state that yields the best overlap between the current scan and the map ϑ_{t-1} at $t-1$.

Implementation Results

The scan matching with particle filter algorithm was successfully implemented on the ER2 robot. Figure 5.7 shows the occupancy grid maps of a 73m x 30m cyclic environment acquired by the robot before and after implementation of the algorithm.

The occupancy grid map in Figure 5.7(a) shows a large loop closure error before implementation of the scan matching with particle filter algorithm. Figure 5.7(b) shows that the loop closure error was reduced tremendously after implementation of the scan matching with particle filter algorithm.



(a) (b)

Figure 5.7: Implementation of the scan matching with particle filter algorithm in a 73m x 30m cyclic environment, Level 3 EA Block of the NUS Engineering Faculty (a) Occupancy grid map shows large loop closure error before scan matching with particle filter. (b) Occupancy grid map shows small loop closure error after scan matching with particle filter.

5.4.2 Loop Closure Detection

The implementation results of the scan matching algorithm discussed in the previous section has shown a significant reduction in the odometry errors. Unfortunately, the odometry errors cannot be completely eliminated with scan matching. The uncertainties associated with scan matching continues to accumulate as the robot moves further and finally manifest itself as a loop closure error when the robot returns to a previously visited location. Hence, an algorithm is needed to detect any loop closure opportunities.

A simple yet effective loop closure detection algorithm is proposed and implemented by the author. This algorithm detects loop closure opportunities by monitoring the uncertainties associated with scan matching. The uncertainties associated with scan matching is estimated by the odometry motion model described in Section 4.3.1. M random samples that represents the uncertainties associated with the scan matching are drawn from state transition probability described by the odometry motion model after every iteration of the scan matching process. Note that the standard deviations σ_1, σ_2, σ_3 and σ_4 for the odometry motion model used to monitor the uncertainties associated with scan matching do not have the same values as the motion model used in localization of the robot in a known environment and scan matching with particle filter. These standard deviations should have smaller values because of the odometry error reduction after scan matching and the values are found experimentally to be $\sigma_1 = 1^o$, $\sigma_2 = 2mm$, $\sigma_3 = 4mm$ and $\sigma_4 = 3^o$.

The experiments to get the standard deviations for the odometry motion model used to monitor the uncertainties associated with scan matching are exactly the same as the experiments to get the standard deviations for the odometry motion model used in localization of the robot in a known environment described in Section 4.3.1 with one subtle difference. The corrected robot poses after the scan matching with particle filter are used as the control data $\mathbf{u_t}$ to compute the fixed intervals of translation and rotation, instead of using the raw odometry readings. As a result, there are smaller discrepancies between the control data and the measured "true" values and hence smaller standard deviations.

A loop closure opportunity is detected when the area covered by the samples that represents the scan matching uncertainty intersects a pose from the trajectory that was previously taken by the robot. Let this pose where the loop closure opportunity is detected be denoted by $\mathbf{x_S}$. Figure 5.8 shows an illustration of a detected opportunity for loop closure. It can be seen that the area covered by samples representing the scan matching uncertainty intersects a pose from the trajectory that was previously taken by the robot.

The loop closure detection is however not robust enough by solely depending on the samples that represents the scan matching uncertainty. False positive will be created if the robot makes an u-turn. Figure 5.9 shows an illustration of a false positive loop closure detection when the robot makes an u-turn. The area covered by samples representing the scan matching uncertainty intersects a pose from the trajectory that

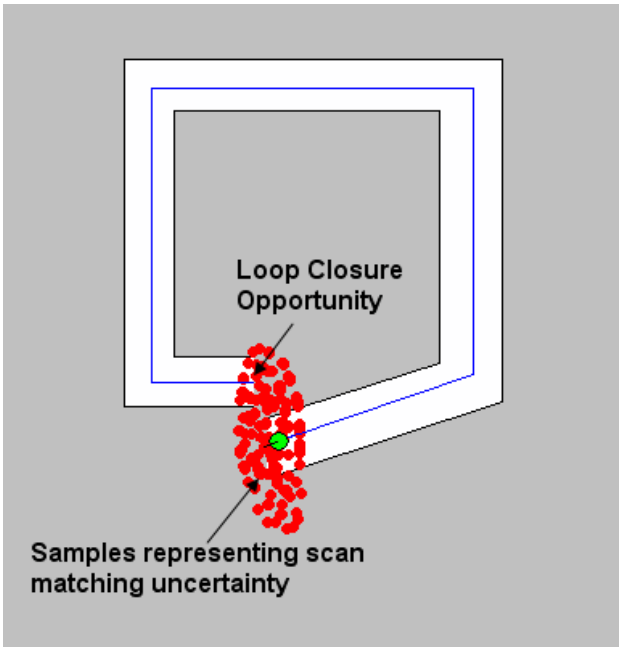


Figure 5.8: An illustration to show that a loop closure opportunity is detected when the area covered by samples representing the scan matching uncertainty intersects a pose from the trajectory that was previously traveled by the robot.

was previously taken by the robot despite that there is no loop.

A topological map is built to prevent false positive loop closure detection [49]. The construction of the topological map starts with adding the first node that corresponds to the starting location of the robot. A new node is added if the distance between the current pose of the robot exceeds a threshold γ from the previous node or if no node is visible from the current pose of the robot. The newly created node is connected to the previous node by an edge. The ray casting operation (see Section 4.3.2) is used to determine if a node is visible from the current robot pose. If a loop closure opportunity was detected by the samples representing the scan matching uncertainty, the number

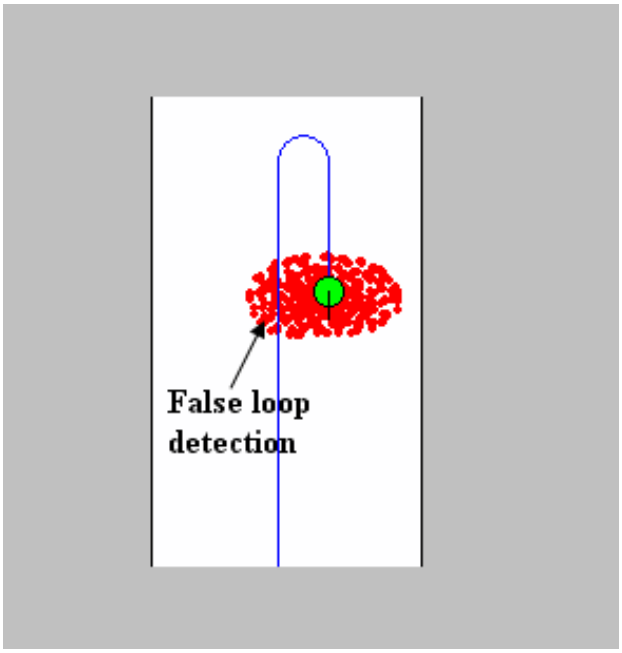
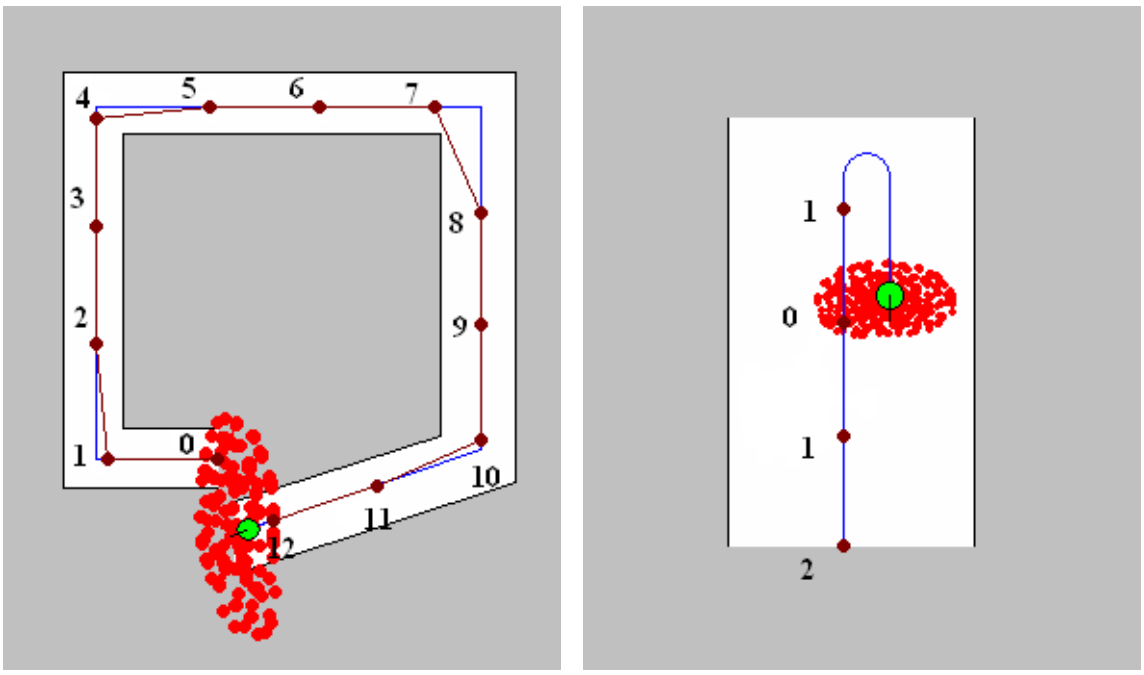


Figure 5.9: An illustration to show that the robot makes a false positive loop closure detection when it makes an u-turn.

of nodes that link the start node (closet node to the current robot pose $\mathbf{x_t}$) and the end node (closest node to $\mathbf{x_S}$) are determined. A positive loop closure opportunity is detected if more than 2 nodes are found in between the start and end nodes. The *Dijkstra's* algorithm [50] is used to count the number of nodes in between the start and end nodes. The algorithm first assigns a value of '0' to the end node. Next, it assigns a value of '1' to all the children of the end node. In general, all children of a parent node with value 'N' are assigned with a value of '$N+1$' until all nodes have been assigned a value. Finally, the total number of nodes in between the start and end nodes are counted following a steepest descent of the node values from the start to end node.



(a) (b)

Figure 5.10: Illustrations of the topological map. The nodes are assigned values according to the *Dijkstra's* algorithm and the number of nodes between the start and end nodes are counted by following the steepest descent (a) A positive loop closure opportunity with 11 nodes in between the start and end nodes. (b) A negative loop closure opportunity with no nodes in between the start and end nodes.

Figure 5.10 show illustrations of the topological map. The nodes are assigned values according to the *Dijkstra's* algorithm and the number of nodes between the start and end nodes are counted by following the steepest descent. Figure 5.10(a) shows a positive loop closure opportunity with 11 nodes in between the start and end nodes and Figure 5.10(b) shows a negative loop closure opportunity with no nodes in between the start and end nodes. Notice that no new nodes are added when the robot makes an u-turn.

Implementation Results

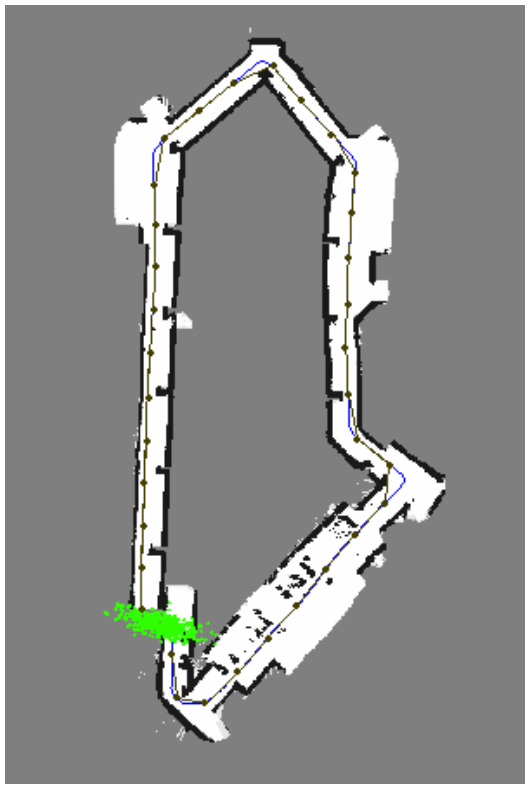


Figure 5.11: Implementation of the loop closure detection algorithm on the ER2 robot. A loop closure opportunity has been detected by the samples representing the scan match uncertainty and confirmed by having more than 2 nodes between the start and end nodes.

The loop closure detection algorithm has been successfully implemented on the ER2 robot. Figure 5.11 shows the implementation results. A loop closure opportunity has been detected by the samples representing the scan match uncertainty and confirmed by having more than 2 nodes between the start and end nodes. Note that the nodes are added at a distance of 5.5m apart.

5.4.3 Loop Closure

Three things need to be done after a loop closure opportunity has been detected. First, the true pose of the robot in the map has to be determined. Let $\mathbf{x_E}$ denote the true pose of the robot. Second, the trajectory of the robot from $\mathbf{x_S}$ to $\mathbf{x_t}$ has to be corrected so as to close the loop. Let this trajectory be denoted by $\mathbf{x_{S:t}}$. Third, the current map ϑ_t has to corrected according to the trajectory of the robot after loop closure.

The true pose of the robot $\mathbf{x_E}$ can be found by using a particle filter. The particles are initialized uniformly within a window centered at $\mathbf{x_S}$. The odometry motion model of this particle filter has the same parameters as the one used for estimating the scan matching uncertainty (see Section 5.4.2). The measurement updates of the particles are done using the current sensor measurement data $\mathbf{z_t}$ and the acquired map ϑ_{t-1} at time $t-1$. The estimated pose from the particles is taken to be $\mathbf{x_E}$ when more than 80 percent of the particles are found within 1m from the pose estimate.

Figure 5.12 shows an illustration of finding $\mathbf{x_E}$ for loop closure. The particles are initialized uniformly within a window centered at $\mathbf{x_S}$ shown in Figure 5.12(b) upon detecting the loop closure opportunity. The particle finally converges to $\mathbf{x_E}$ as the robot moves from Figure 5.12(c) to 5.12(d).

The loop can be closed with the knowledge of both $\mathbf{x_S}$ and $\mathbf{x_E}$. A *forward-backward pose correction* algorithm [51] is adopted for loop closure. Assuming that $\mathbf{x_S}$ and $\mathbf{x_E}$ are error free poses of the robot at the start and end of the loop respectively, the

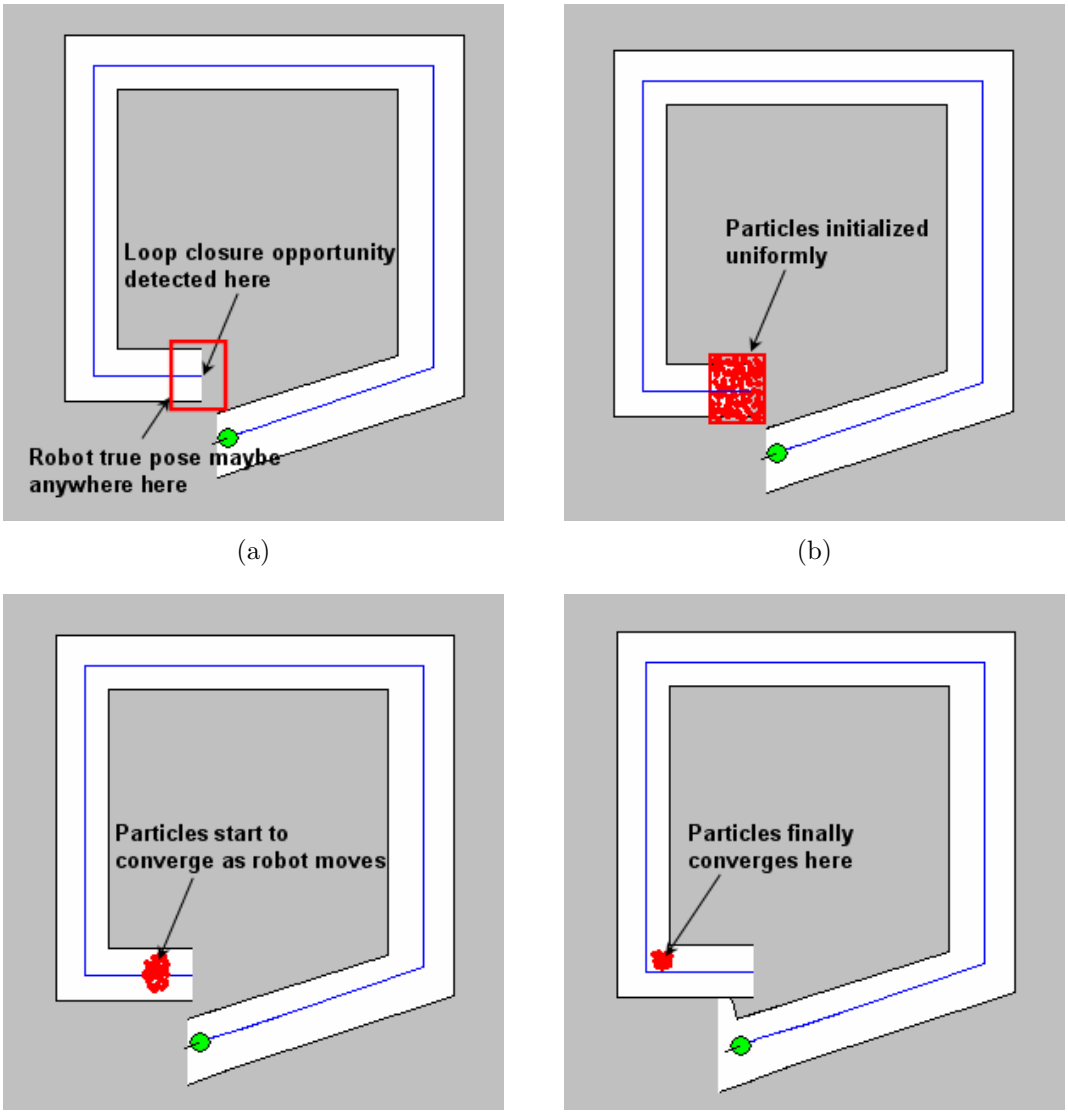


Figure 5.12: Illustrations of finding $\mathbf{x_E}$ for loop closure. (a) $\mathbf{x_E}$ maybe anywhere within a window centered at $\mathbf{x_S}$. (b) The particles are initialized uniformly within the window. (c) Particles start to converge as robot moves. (d) The pose estimate is taken to be $\mathbf{x_E}$ when more than 80 percent of the particles are found within 1m from the pose estimate.

corrected trajectory $\mathbf{x}_{\mathbf{S:t}}^{\mathbf{corrected}}$ after loop closure is given by weighted average of the forward $\mathbf{x}_{\mathbf{S:t}}^{\mathbf{forward}}$ and backward $\mathbf{x}_{\mathbf{S:t}}^{\mathbf{backward}}$ trajectories shown in Equation 5.16.

$$\mathbf{x}_{\mathbf{k}}^{\mathbf{corrected}} = \alpha_k \mathbf{x}_{\mathbf{k}}^{\mathbf{forward}} + (1 - \alpha_k)\mathbf{x}_{\mathbf{k}}^{\mathbf{backward}}, \quad \text{for k = S, S+1, ..., t} \tag{5.16}$$

The forward trajectory $\mathbf{x}_{\mathbf{S:t}}^{\mathbf{forward}}$ is the trajectory of the robot propagated forward from $\mathbf{x_S}$ to $\mathbf{x_t}$ and is given by

$$\mathbf{x}_{\mathbf{k}}^{\mathbf{forward}} = \mathbf{x_k}, \quad \text{for k = S, S+1, ..., t} \tag{5.17}$$

and the backward trajectory $\mathbf{x}_{\mathbf{S:t}}^{\mathbf{backward}}$ is trajectory of the robot propagated backward from $\mathbf{x_E}$ to $\mathbf{x_S}$ and is given by

$$\mathbf{x}_{\mathbf{k}}^{\mathbf{backward}} = \begin{cases} \mathbf{x_E}, & \text{if k = t} \\ \mathbf{x}_{\mathbf{k+1}}^{\mathbf{backward}} + \mathbf{x_k} - \mathbf{x_{k+1}}, & \text{if k = t-1, t-2, ..., S} \end{cases} \tag{5.18}$$

α_k is the weighing factor for the forward and backward trajectories and is given by Equation 5.13(d). Notice that $\alpha_k = 1$ and $(1 - \alpha_k) = 0$ when k = S. As a result, from Equation 5.16 $\mathbf{x}_{\mathbf{S}}^{\mathbf{corrected}} = \mathbf{x_S}$. This result is reasonable because $\mathbf{x_S}$ is taken to be a error free pose at k = S. Similarly, $\mathbf{x}_{\mathbf{t}}^{\mathbf{corrected}} = \mathbf{x_E}$ because $\mathbf{x_E}$ is taken to be a error free pose at k = t.

$$\alpha_k = \frac{t - k}{t - S}, \quad \text{for k = S, S+1, ..., t} \tag{5.19}$$

Finally, the full trajectory of the robot $\mathbf{x_{1:t}}$ is given by

$$\mathbf{x_k} = \begin{cases} \mathbf{x_k}, & \text{for k = 1,2, ..., S-1} \\ \mathbf{x}_{\mathbf{k}}^{\mathbf{corrected}}, & \text{for k = S, S+1, ..., t} \end{cases} \tag{5.20}$$

Figure 5.13 shows a simulation of the forward-backward loop closure algorithm. Figure 5.13(a) shows a detected loop closure opportunity. Figure 5.13(b)shows the forward trajectory obtained from Equation 5.17 and Figure 5.13(c) shows the backward trajectory obtained from Equation 5.18. The loop is closed in Figure 5.13(d). Notice that $\mathbf{x_S}$ and $\mathbf{x_E}$ remains the same because they are taken to be error free pose.

The current map ϑ_t can be corrected according to the trajectory of the robot after loop closure. This is done by removing the current map ϑ_t and replacing it with a new map generated by the occupancy grid mapping algorithm described in Section 5.3 with the corrected trajectory of the robot and the laser sensor measurements associated with each robot pose from the corrected trajectory as the inputs.

Implementation Results

The loop closure algorithm has been successfully implemented on the ER2 robot. Figure 5.14 shows the results for the implementation of finding $\mathbf{x_E}$ for loop closure. Figure 5.14(a) shows the initialization of the particles uniformly within the window centered at $\mathbf{x_S}$. $\mathbf{x_E}$ maybe anywhere within this window. The particles eventually converges to $\mathbf{x_E}$ as the robot moves from Figure 5.14(b) to Figure 5.14(c).

Figure 5.15 shows the occupancy grid map after the loop closure using the forward-backward loop closure algorithm. Notice that the samples representing the uncertainty associated with scan matching are shifted to the $\mathbf{x_E}$ with smaller uncertainty because $\mathbf{x_E}$ is taken to be an error free pose. The robot is traveling in previously visited locations after the loop closure and the standard deviations σ_1, σ_2, σ_3 and σ_4

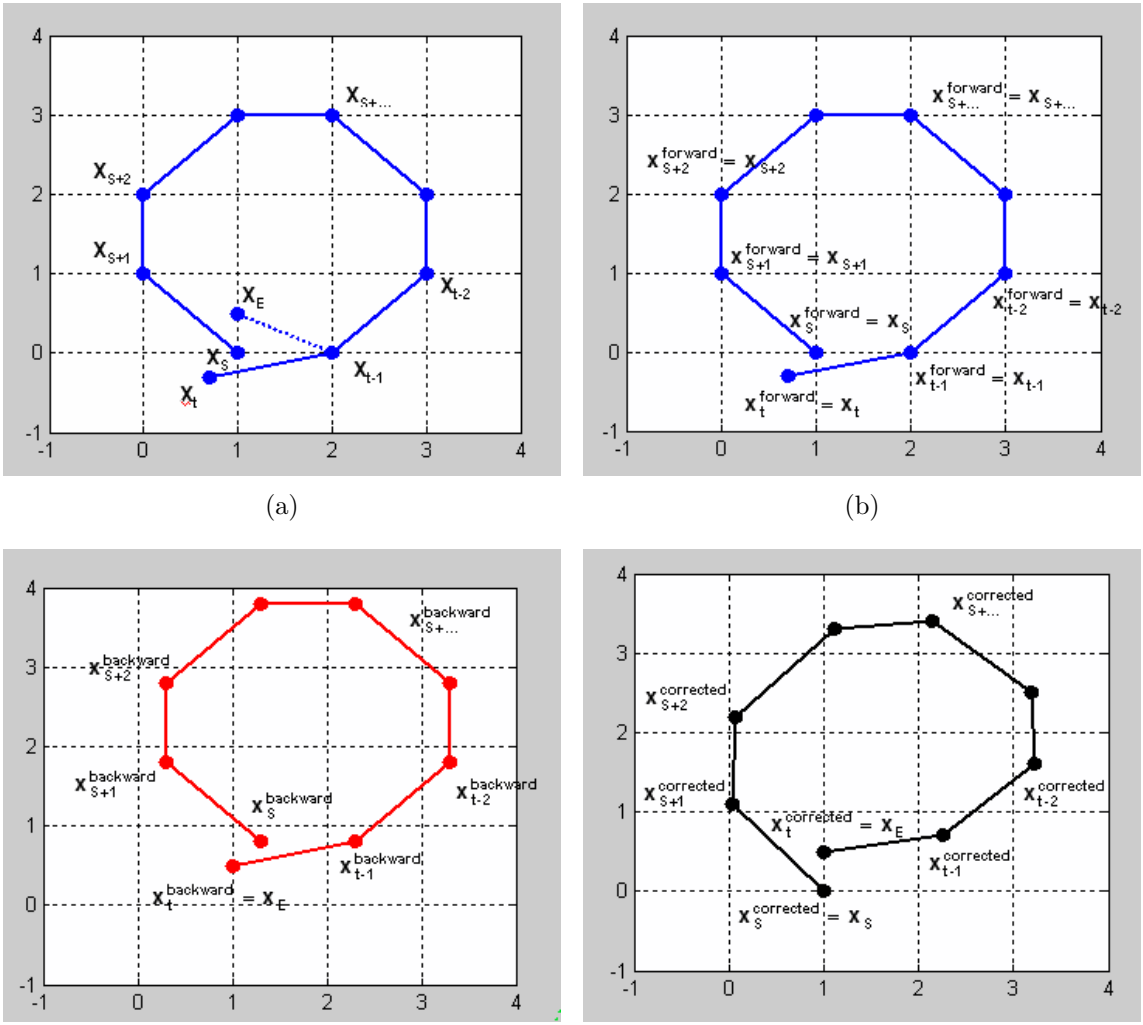
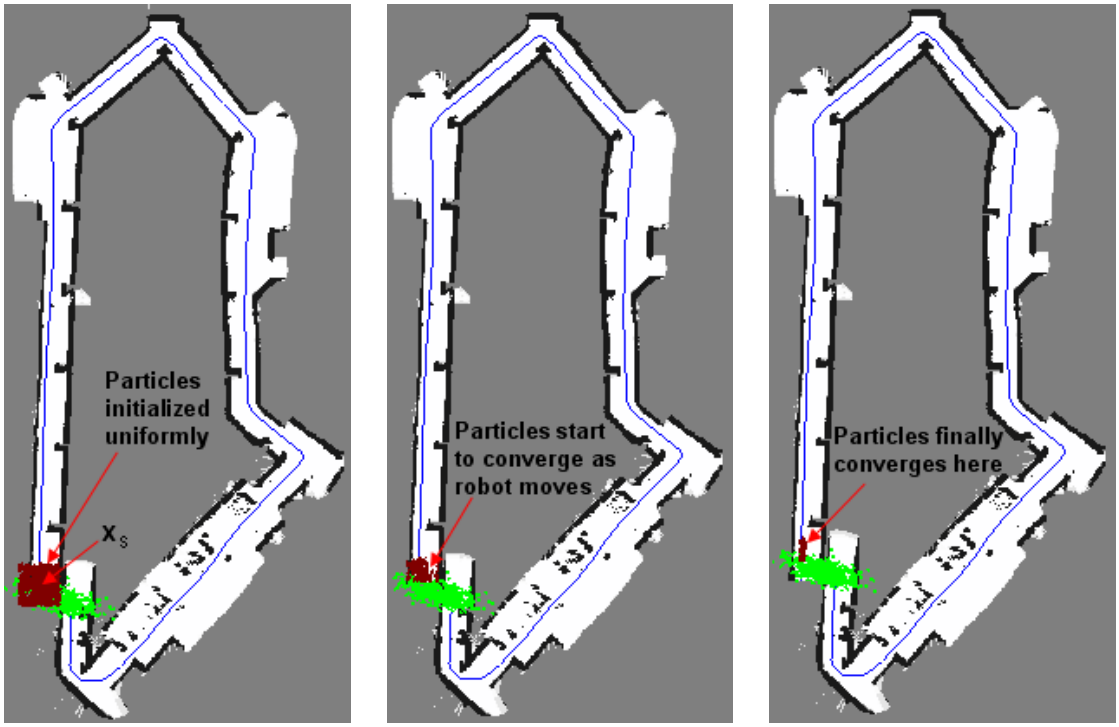


Figure 5.13: Simulation of the forward-backward loop closure algorithm. (a) Loop closure opportunity detected. (b) Forward trajectory where $\mathbf{x_S^{forward} = x_S}$. (c) Backward trajectory where $\mathbf{x_t^{backward} = x_E}$. (d) Corrected trajectory where $\mathbf{x_S^{corrected} = x_S}$ and $\mathbf{x_t^{corrected} = x_E}$.

from the odometry motion model used to model the uncertainty associated with scan matching should have smaller values than the standard deviations from the odometry motion model used to model the uncertainty associated with the scan matching in unknown locations (see Section 5.4.2). This is because the current scan is matched with a full map of the previously visited environment hence higher accuracy. These standard deviations are experimentally found to be $\sigma_1 = 0.6^o$, $\sigma_2 = 0.9mm$, $\sigma_3 = 1.3mm$ and $\sigma_4 = 1.9^o$.

The experiments to get the values for standard deviations are the same as the experiments to get the standard deviations for the odometry motion model used to model the uncertainty associated with the scan matching in unknown locations described in Section 5.4.2 except for one difference. The corrected robot poses from the scan matching with particle filter against a fully known map are used as the control data $\mathbf{u_t}$ to compute the fixed translation and rotation intervals, instead of using the corrected robot poses after the scan matching with particle filter against a partially known map. As a result of the scan matching against a fully known map, there are smaller discrepancies between the control data and the measured "true" values and hence smaller standard deviations.



(a) (b) (c)

Figure 5.14: Results for the implementation of finding $\mathbf{x_E}$ for loop closure. (a) The initialization of the particles uniformly within the window centered at $\mathbf{x_S}$. The particles eventually converges to $\mathbf{x_E}$ as the robot moves from (b) to (c).

Figure 5.15: Occupancy grid map of Level 3 EA Block of NUS Engineering Faculty after loop closure.

CHAPTER 6

CONCLUSION

6.1 Summary

Localization is the process of determining the pose of a mobile robot with respect to a given map of the environment (known environment). It is the most fundamental and important problem in mobile robotics. This is because a mobile robot must know its pose at every instance of time so as to determine its destination and plan a path that will enable it to navigation there safely. The localization problem can be made more difficult in cases where the map of the environment is not given (unknown environment) to the robot. This is the SLAM problem where the robot has to simultaneously build a map of its environment and localizes itself with respect to this map.

The main objective of this thesis is to investigate and implement algorithms to estimate the robot state $\mathbf{x_t}$ at every instance of time in both known and unknown indoor environments. Probabilistic methods are selected over other deterministic methods because it is more robust to represent $\mathbf{x_t}$ by probability distributions over a whole space of guesses than relying on a single "best guess". In general, a probabilistic solution is needed to estimate $\mathbf{x_t}$ recursively.

In Chapter 2, some of the terms that are commonly used in the context of probabilistic mobile robot localization were defined. These terms include *state*, *sensor measurements*, *control actions*, *belief distributions*, *state transition* and *measurement probabilities*. The Bayes filter, which is the most general form of probabilistic recursive state estimation, was reviewed.

In Chapter 3, detailed analysis of the characteristics, advantages and disadvantages were given for some sensors that are commonly used for the localization of a mobile robot. These sensors include IMU, compass, GPS, odometer, Cricket Motes, NorthStar localization kit and laser range finder. The odometer and laser range finder were found suitable for mobile localization in both known and unknown indoor environments. The two sensors were used for the implementation of the localization algorithms in this dissertation.

In Chapter 4, the particle filter was discussed in detail. The particle filter is a practical variation of the Bayes filter that seeks to represent the belief distribution with a finite number of samples known as the particles. Three different variations of the particle filter were reviewed for solving three localization problems in a known environment. They are local localization, global localization and the kidnapped problem. The localization algorithms using particle filter were successfully simulated in virtual environments and implemented on the ER2 robot.

The most significant contribution of this thesis is found in Chapter 5. In this chapter, a complete solution for the simultaneous localization and mapping of a mobile

robot in an unknown environment was proposed. This novel SLAM algorithm uses a laser scan matching algorithm to align consecutive laser scans, loop closure detection algorithm to detect loop closure opportunity and loop closure algorithm to close any detected loops in the map. The SLAM algorithm was successfully implemented on the ER2 robot.

6.2 Further Works

Some of the further works to improve the contributions are as follows:

1. Only a few key sensors commonly used for robot localization could be evaluated due to the time constrain of this dissertation. Further works can be done to include more sensors in the evaluation. Examples of other sensors are inclinometers, bumper switches and wheel encoders etc.

2. A key limitation of the localization algorithms for both known and unknown (SLAM) environments implemented in this research is that they may fail in highly dynamic environments. This is because the sensor measurement model used in this research does not account for corruption of the sensor measurement data by the state of dynamic objects in the environment. Further works can be done to improve the sensor measurement model so that it is able to tolerate corruptions from dynamic objects in the environment.

3. For a mobile robot to be truly autonomous, it must be able to acquire maps of unknown environments, navigate from a point of origin to a destination using this map, and determine its pose with respect to the acquired map at all times.

The SLAM algorithm proposed and implemented in this thesis allows the robot to acquire maps of unknown environments and determine its pose with respect to this map at all times. However, it does not allow the robot to navigate from a point of origin to a destination using acquired map. Further works can be done to integrate navigation algorithms into the SLAM algorithm to make the robot truly autonomous. Examples of navigation algorithms are the *navigation functions* [23], *artificial potential field* [52], *vector field histogram* [53], *hybrid navigation algorithm* [54] and the *integrated algorithm*[55].

4. Implementation of the localization algorithms are carried out in only indoor environments. The sensors are also selected for indoor environments. Further works can be carried test the robustness of the algorithms in outdoor environments. Sensors such as compass, GPS and IMU [26] can be used for mobile robot localization in the outdoor environments.

BIBLIOGRAPHY

[1] John J. Leonard and Hugh F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," in *IEEE Transaction on Robotics and Automation*, June 1991, vol. 7, pp. 376–382.

[2] B. Everett J. Borenstein and L. Feng, *Navigating Moblie Robots: Systems and Techniques*, Wellesley, MA: A.K.Peters, Ltd, 1996.

[3] J. Borenstein and L. Feng, "Measurement and correction of systematic odometry errors in mobile robots," in *IEEE Transaction on Robotics and Automation*, December 1996, vol. 12, pp. 869–880.

[4] J. Borenstein and L. Feng, "Umbmark: A benchmark test for measuring odometry errors in mobile robots," in *Proceedings of the 1995 SPIE Conference on Mobile Robots*, October 1995, pp. 569–574.

[5] J. Borenstein and Y. Koren, "Herarchial computer system for autonomous vehicle," in *Proceedings of the 8th Israelic Convention on CAD/CAM and Robotics*, December 1986.

[6] C. Cohen and F. Koss, "A comprehensive study of three object triangulation," in *Proceedings of the 1993 SPIE Conference on Mobile Robots, Bosten, MA*, November 1992, pp. 95–106.

[7] MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139, *Cricket V2 User Manual*, 2 edition, July 2004.

[8] Evolution Robotics, Inc., 130 W. Union Street, Pasadena, CA 91103, *NorthStar Projector Kit User Guide*, 1.0 edition.

[9] Evolution Robotics, Inc., 130 W. Union Street, Pasadena, CA 91103, *NorthStar Detector Kit User Guide*, 2.1 edition.

[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2005.

[11] Martin David Adams, *Sensor Modelling, Design and Data Processing for Autonomous Navigation*, vol. 13, World Scientific, 1999.

[12] Schiele B. and J.L. Crowley, "A comparison of position estimation techniques using occupancy grids," in *Proceedings of the International Joint Conference on Robotics and Automation (ICRA)*, 1999.

[13] Konolige K. and K. Chou, "Markov localization using correlation," in *Proceedings of the International Joint Conference on Artificial Intelligencce (IJCAI)*, 1999.

[14] C. Kwok, D. Fox, and M. Meila, "Real-time particle filters," in *Proceedings of the IEEE*, March 2004, vol. 92, No. 3.

[15] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," in *Artificial Intelligence Journal (AIJ)*, 2001.

[16] D. Fox C. Kwok and M. Meila, "Adaptive real-time particle filters," in *Proceedings of the International Joint Conference on Robotics and Automation (ICRA)*, 2003.

[17] Ioannis M. Rekleitis, "A particle filter tutorial for mobile robot localization," Tech. Rep. TR-CIM-04-02, Centre for Intelligent Machines, McGill University, 3480 University St., Montreal, Québec, CANADA H3A 2A7, 2004.

[18] Hugh Durrant-Whyte and Tim Bailey, *Simultaneous Localization and Mapping: Part I*, IEEE Robotics and Automation Magazine, June 2006.

[19] Hugh Durrant-Whyte and Tim Bailey, *Simultaneous Localization and Mapping: Part II*, IEEE Robotics and Automation Magazine, June 2006.

[20] Paul Newman and Kin Ho, "Slam-loop closing with visually salient features," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 18-22 April 2005.

[21] Cheeseman P. and P. Smith, "On the representation and estimation of spatial uncertainty," in *International Journal of Robotics*, 1986, pp. 5:56–68.

[22] M. Montemerlo, S. Thrun, D. Koller, and B. Webreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the AAAI National Conference of Artificial Intelligence*, 2002, pp. 593–598.

[23] J.C. Latombe, *Robot Motion Planning*, Boston : Kluwer Academic Publishers, 1991.

[24] Richard A. Johnson, *Probability and Statistics for Engineers*, Prentice Hall, 6 edition, 2000.

[25] Ahmed El-Rabbany, *Introduction to GPS : the global positioning system*, Boston, Mass. : Artech House, 2002.

[26] S. Sukkarieh, E.M. Nebot, and H. Durrant-Whyte, "A high integrity IMU/GPS navigation loop for autonomous land vehicle applications," in *IEEE Transactions on Robotics and Automation*, June 1999, vol. Vol 15, No 3.

[27] Crossbow Technology, Inc, 4145 N. First Street, San Jose, CA 95134, *RGA Series User's Manual*, revision a edition, March 2005.

[28] Roland Siegwart and Illah Nourbakhsh, *Introduction to Autonomous Mobile Robot*, Intelligent Robotics and Automation Agents. The MIT Press, Massachusetts Instittute of Technology Cambridge.

[29] Honeywell, Solid State Electronics Center, 12001 Highway 55, Plymouth MN 55441, *HMR3200/HMR3300 Digital Compass Solutions User's Guide*, 04-02 revision a edition.

[30] Myke Predko, *Programming Robot Controllers*, McGraw Hill, 2003.

[31] Nissanka Bodhi Priyantha, *The Cricket Indoor Location System*, Ph.D. thesis, Massachusetts Institute of Technology, June 2005.

[32] Hokuyo Automatic Co. Ltd., Kokudo Building 2-1-12, Sonezaki, Kita-ku, Osaka 530-0057, Japan, *URG Series Communication Protocol Series*, 1 edition, Feb 2004.

[33] Sen Zhang, Lihua Xie, Martin Adams, and Fan Tang, "Geometrical feature extraction using 2d range scanner," in *The Fourth International Conference on Control and Automation (ICCA03), Montreal, Canada*, 10-12 June 2003.

[34] G. Welch and G.Bishop, "An introduction to the kalman filter," Tech. Rep. TR 95-041, Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599-3175, February 1995.

[35] Jodo Xavier, Daniel Castrot, Marco Pachecot, Ant Nio Ruanot, and Urbano Nunes, "Fast line, arc/circle and leg detection from laser scan data in a player driver," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation Barcelona, Spain*, April 2005, pp. 95–106.

[36] H. Choset, *Sensor based motion planning: The hierarchical generalized voronoi graph*, Ph.D. thesis, California Institute of Technology, 1996.

[37] H. Choset and J.W. Burdick, "Sensor based planning: The hierarhical generalized voronoi graph," in *Proceeding Workshop on Algorithmic Foundations of Robotics, Toulouse, France*, 1996.

[38] D. Kortenkamp and T.Weymouth, "Topological mapping for mobile robots using a combination of sonar and vision sensing," in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, July 1994, pp. 979–984.

[39] Yamauchi B. and P.Langley, "Place recognition in dynamic environments," in *Journal of Robotic Systems*, 1997.

[40] William M. Newman and Robert F. Sproull, *Principles of Interative Computer Graphics*, MacGraw-Hill Internation Series, 2 edition, 1979.

[41] H.P. Moravec, "Sensor fusion in certainty grids for mobile robots," in *AI Magazine*, Summer 1988, pp. 61–74.

[42] H.P. Moravec, "A bayesian method for certainty grids," in *AAAI 1989 Spring Symposium on Mobile Robots*, 1989.

[43] A. Elfes, "Sonar-based real-world mapping and navigation," in *IEEE Journal of Robotics and Automation*, June 1987, pp. 249–265.

[44] A. Elfes, *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*, Ph.D. thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1989.

[45] P.J. Besl and N.D. MacKay, "A method for registration of 3d shapes," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992, pp. 239–256.

[46] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," in *Journal of Intelligent and Robotic Systems*, 1997, pp. 249–257.

[47] G. Wei, C. Wetzler, and E. von Puttkamer, "Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans," in *Proceedings of the International Conference on Intelligent Robots and Systems*, 1994, pp. 595–601.

[48] A. Diosi and L. Kleeman, "Laser scan matching in polar coordinates with application to slam," in *Proceedings of the International Conference on Intelligent Robots and Systems*, August 2005, pp. 3317– 3322.

[49] C. Stachniss, D. Hähnel, W. Burgard, and G. Grisetti, "On actively closing loops in grid-based fastslam," *Advanced Robotics - The Int. Journal of the Robotics Society of Japan (RSJ), 2005*, vol. 19, no. 10, pp. 1059–1080, 2005.

[50] Erwin Kreyszig, *Advance engineering mathematics*, John Wiley and SONS, INC., 8 edition, 1999.

[51] S. Hagen and B.Krose, "Trajectory reconstruction for self-localization and map building," in *Proceedings of the International Conference on Robotics and Automation*, 2002, pp. 1796–1801.

[52] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *International Journal of Robotic Research*, 1986, vol. 5, pp. 90–98.

[53] J.Borenstien and Y.Koren, "The vector field histogram - fast obstacle avoidance for mobile robots," in *IEEE Journal of Robotics and Automation*, June 1991, vol. 7, pp. 278–288.

[54] Lim Chee Wang, Lim Ser Yong, and Marcelo H. Ang Jr, "Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment," in *Proceedings of the IEEE International Symposium on Intelligent Control, Vancouver Canada*, 2002, pp. 821–826.

[55] Lee Gim Hee, Lim Chee Wang, and Marcelo H. Ang Jr, "An integrated algorithm for autonomous navigation of a mobile robot in an unknown environment," in *Third Humanoid, Nanotechnology, Information Technology, Communication and Control Environment and Management (HNICEM) International Conference*, March 2007.