Autonomous Mobile Robot Positioning using

Unscented HybridSLAM

by

Amirhossein Monjazeb


A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Ottawa-Carleton Institute for Mechanical and Aerospace Engineering

Department of Mechanical and Aerospace Engineering

Carleton University

Ottawa, Ontario

Canada

November, 2013

© Copyright 2013

Amirhossein Monjazeb
Abstract

Simultaneous localization and mapping (SLAM) is a process in which a mobile robot travels through an environment and concurrently makes a momentary map of the environment and uses that map to localize itself. The simultaneous localization and mapping is currently one of the most challenging problems in the field of autonomous mobile robots and providing a solution to SLAM may open doors to the world of truly autonomous robots. The most significant contribution of this dissertation is to provide a novel approach to Simultaneous Localization and Mapping problem in extensive outdoor environments and based on estimation approach. The new approach is called Unscented HybridSLAM filter which presents a consistent mathematical model out of a rigorous probabilistic Bayesian-based framework. It is theoretically proven that the map converges and how the new approach can handle correlations that arise between error in motion and error in observation. It is also shown that there is no need for a large storage of information since the inherent structure of Unscented HybridSLAM does not require memory as much as its counterpart filters. The map evolution of the new algorithm is examined in detail as well as its performance. The new approach is compared to currently used algorithms in particular EKF-SLAM, FastSLAM, and HybridSLAM and results are probed and discussed in different simulated scenarios. Together, the theoretical modeling and simulations results prove the consistency of Unscented HybridSLAM and show that it is possible to apply Unscented HybridSLAM as an alternative algorithm for real implementations.
Acknowledgment

I would like to thank all of the people who encouraged and supported me during the undertaking of this research. To begin, I would like to express my gratitude to my Ph.D. program advisor, Professor Jurek Z. Sasiadek, for his continued support and guidance. He provided advice, opportunities, and assistance that greatly enhanced my research experience. I will be forever thankful to Dr. Vladimir Polotski for his motivation, and encouragement to move forward.

I want to say thanks to Dr. Michael Montemerlo from Carnegie Mellon University, whose work was the best inspiration to this research and for his supporting words and emails. Special thanks to Professor Hugh Durrant-Whyte, Professor Eduardo Nebot, and Dr. Tim Bailey from Australian Centre for Field Robotics in Sydney, who patiently answered my research questions.

I am deeply thankful to my Mother and Father whose support and care were encouraging me during this work and to my beloved sister Golchehr who was always available for reviewing my work and was with me to the end. Last but not certainly least, I would like to thank my lovely wife Lida whose love and patience was the most valuable support to me.

Online Support

A free software code for robotics research intentions was used in this dissertation. The software in form of both MATLAB and C++ codes is available at http://www.lasmea.univ-bpclermont.fr/ftp/pub/trassou/SLAM/SLAM_Summer_School2002/SLAM%20Summer%20School%202002.htm
Contents

Abstract i
Acknowledgement ii
Contents iii
Notation viii
Acronyms xii

1. Introduction 1

1.1 Simultaneous Localization and Mapping 3
1.2 SLAM Applications 4
1.3 Feature-based SLAM 6
1.4 Probabilistic SLAM 7

1.4.1 Probability Distribution 7
1.4.2 Process Models 8
1.4.2.1 Modeling Sensor 8
1.4.2.2 Modeling Motion 9
1.4.3 Bayesian SLAM 9
1.4.4 Belief 10
1.4.4.1 Prior and Posterior Beliefs 11
1.4.4.2 Total Probability and Markov Localization 12

1.5 Main Contribution of this Thesis 14

1.6 Thesis Overview 15
2. Problem Description

2.1 Problem Definition

2.2 Uncertainty

2.2.1 Uncertainty Analysis

2.2.2 Uncertainty in Motion

2.2.3 Uncertainty in Observation

2.3 Currently Used Algorithms

2.3.1 EKF-SLAM

2.3.1.1 Computational Complexity

2.3.1.2 Single Hypothesis Property

2.3.1.3 Noise Characteristics

2.3.1.4 System Nonlinearity

2.3.2 SLAM based on Rao-Blackwellised Particle Filter

2.3.2.1 Recursive Monte Carlo Sampling

2.3.2.2 Sample Impoverishment

2.3.2.3 Overconfidence

2.3.3 HybridSLAM

2.3.3.1 Modification

2.3.3.2 Diversity in Sample Set

2.4 Comparison of UHS with Currently Used Methods

2.5 Summary
3. Unscented HybridSLAM

3.1 Estimation of the Joint Posterior ................................................. 40
   3.1.1 Factorization ................................................................. 41
   3.1.2 Derivation ................................................................. 42
3.2 Single Gaussian Probability Function ........................................ 45
3.3 Function Linearization ............................................................ 47
   3.3.1 Perturbation ................................................................. 47
   3.3.2 Linearization in UHS ...................................................... 48
   3.3.3 UKF Algorithm ............................................................. 49
3.4 Constrained Local Sub-Map Filter ............................................. 52
   3.4.1 Uncertainty in Position and Observation of Landmarks .............. 54
   3.4.2 Reducing Uncertainty ..................................................... 56
   3.4.3 Adding Landmarks to the Local Map .................................. 57
   3.4.4 State and Covariance Matrices after Fusion .......................... 59
3.5 Solution to Sample Impoverishment ......................................... 61
   3.5.1 Proposal Distribution .................................................... 62
   3.5.2 Updating Importance Weight ........................................... 67
3.6 Data Association ................................................................. 70
   3.6.1 Data Association for Each Particle .................................... 70
   3.6.2 Maximum Likelihood Estimator ........................................ 71
   3.6.3 Elimination of Wrong Observation .................................... 72
3.7 Logarithmic Computational Complexity .................................... 74
3.8 Proof of Filter Convergence ................................................... 76
5. Results and Discussion

5.1 Performance Comparison

5.1.1 Gaussian Implication

5.1.2 Nearby Landmarks

5.1.3 Observation and Motion Noise

5.1.4 Process Time

5.2 Loop Closing

5.2.1 Consistency

5.2.2 Loop Size

5.3 Evolution of the Map

5.3.1 Robustness

5.3.2 Map Fusion

5.3.3 Map Update

5.4 Distribution of Landmarks

5.4.1 Uniform Distribution of Landmarks

5.4.2 Random Distribution of Landmarks

5.4.2.1 Threshold

5.4.2.2 Double Loop

5.5 Future Work

6. Conclusion

References

Appendices
## Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_k$</td>
<td>set of all data association</td>
</tr>
<tr>
<td>$d_{k,i}$</td>
<td>data association for landmark $i$</td>
</tr>
<tr>
<td>$\hat{d}_{k,i}$</td>
<td>maximum likelihood</td>
</tr>
<tr>
<td>$d_i$</td>
<td>partial first order difference</td>
</tr>
<tr>
<td>$d_i^2$</td>
<td>partial second order difference</td>
</tr>
<tr>
<td>$D$</td>
<td>distance between sensor and landmark</td>
</tr>
<tr>
<td>$\bar{D}_{\Delta x}$</td>
<td>first order central divided difference operator</td>
</tr>
<tr>
<td>$\bar{D}_{\Delta x}^2$</td>
<td>second order central divided difference operator</td>
</tr>
<tr>
<td>$e_i$</td>
<td>$i^{th}$ unit vector</td>
</tr>
<tr>
<td>$E[\cdot]$</td>
<td>expected value</td>
</tr>
<tr>
<td>$f(\cdot)$</td>
<td>non-linear motion function</td>
</tr>
<tr>
<td>$h(\cdot)$</td>
<td>non-linear sensor function</td>
</tr>
<tr>
<td>$k$</td>
<td>discrete time index</td>
</tr>
<tr>
<td>$K_k$</td>
<td>Kalman filter gain</td>
</tr>
<tr>
<td>$\ell$</td>
<td>wheel to wheel distance</td>
</tr>
<tr>
<td>$\ell$</td>
<td>interval length or central difference step size</td>
</tr>
<tr>
<td>$L^S$</td>
<td>location of center of range sensor</td>
</tr>
<tr>
<td>$L^C$</td>
<td>location of middle point of the back axel of the robot</td>
</tr>
</tbody>
</table>
L  dimension of a state
m  set of all landmarks
m_i  vector of true location of the i^{th} landmark
m_i  mean operator
N(\mu, \sigma)  normal distribution with mean \mu and covariance \sigma
P_k^{-1} (aug)  augmented form of system covariance
P_k^+  posterior covariance matrix of the system
P_{x_k|z_k}  state observation cross covariance matrix
P_x  covariance matrix of x
P(\ldots | \ldots)  conditional probability
Q_k  system noise covariance matrix
R_k  observation noise covariance matrix
S_x  Cholesky factor
s_{x_i}  i^{th} column of the Cholesky factor of covariance matrix of x
T_p  transformation vector
u_k  control vector applied at time
v_k  observation noise
V_R  robot linear velocity
V^R  velocity of the middle back axle of the robotic vehicle
V^e  velocity of the vehicle measured by the encoder reader
w_k  system noise
\( w^R_k \)  
motion noise

\( w^u_k \)  
control noise

\( \tilde{w}^n_k \)  
importance weight

\( x^-_k \)  
prior state

\( x^+_k \)  
posterior state

\( x^R_k \)  
state of the robot at time step \( k \)

\( x^m \)  
vector of all landmarks locations

\( x^R_k^n \)  
state of the robot for particle \( n \)

\( x^S \)  
sensor pose

\( \bar{x} \)  
prior mean of \( x \)

\( x_{k-1} (\text{aug}) \)  
augmented form of estimated state

\( X^R_k \)  
path of the robot up to time step \( k \)

\( x^R_k^n \)  
path of the robot for particle \( n \)

\( X^S \)  
sensor path

\( y \)  
\( i^{th} \) component of \( y - \bar{y} \) \( (i = 1, \ldots, L) \)

\( z_{k,i} \)  
observation of the \( i^{th} \) landmark

\( z_k \)  
generic observation of one or more landmarks

\( \tilde{z}^-_k \)  
prediction observation

\( \tilde{z}_k \)  
innovation

\( \tilde{Z}_k \)  
innovation covariance matrix
\( \mathbf{Z}_k \) set of all observations

\( \alpha \) sigma points constant around the augmented state of the system

\( \beta \) sensor bearing

\( \beta \) Gaussian distribution of the system incorporate factor

\( \kappa \) second scaling parameter

\( \Delta \) mean square

\( \Delta_y \) zero-mean unity variance random variable

\( \varphi \) angle between robot’s heading and the global reference system

\( \Phi_k \) matrix of the partial derivatives of system function

\( \Psi_k \) matrix of partial derivatives of the observation function
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Bel$</td>
<td>Robot Belief</td>
</tr>
<tr>
<td>$Bel^-$</td>
<td>Robot Prior Belief</td>
</tr>
<tr>
<td>$Bel^+$</td>
<td>Robot Posterior Belief</td>
</tr>
<tr>
<td>CDA</td>
<td>Constraint Data Association</td>
</tr>
<tr>
<td>CDF</td>
<td>Central Difference Filter</td>
</tr>
<tr>
<td>CEKF</td>
<td>Compressed Extended Kalman Filter</td>
</tr>
<tr>
<td>CF</td>
<td>Cholesky Factor</td>
</tr>
<tr>
<td>CLSF</td>
<td>Constraint Local Sub-map Fusion</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DBN</td>
<td>Dynamic Bayesian Network</td>
</tr>
<tr>
<td>DDF</td>
<td>Divided Deference Filter</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>GM</td>
<td>Global Map</td>
</tr>
<tr>
<td>GRV</td>
<td>Gaussian Random Variable</td>
</tr>
<tr>
<td>HS</td>
<td>HybridSLAM</td>
</tr>
<tr>
<td>ICR</td>
<td>Instant Center of Rotation</td>
</tr>
<tr>
<td>ISU</td>
<td>Importance Sampling Update</td>
</tr>
<tr>
<td>IW</td>
<td>Importance Weight</td>
</tr>
<tr>
<td>JCBB</td>
<td>Joint Compatibility Branch and Bound</td>
</tr>
<tr>
<td>KF</td>
<td>Kalman Filter</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Term</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>KG</td>
<td>Kalman Gain</td>
</tr>
<tr>
<td>LEU</td>
<td>Landmark Estimation Update</td>
</tr>
<tr>
<td>LQE</td>
<td>Linear Quadratic Estimation</td>
</tr>
<tr>
<td>LMS</td>
<td>Least Mean Squares</td>
</tr>
<tr>
<td>MCL</td>
<td>Monte Carlo Localization</td>
</tr>
<tr>
<td>ML</td>
<td>Markov Localization</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum Likelihood Estimator</td>
</tr>
<tr>
<td>OE</td>
<td>Orientation Error</td>
</tr>
<tr>
<td>PF</td>
<td>Particle Filter</td>
</tr>
<tr>
<td>R-B</td>
<td>Rao-Blackwellization</td>
</tr>
<tr>
<td>RBPF</td>
<td>Rao-Blackwellised Particle Filter</td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Square</td>
</tr>
<tr>
<td>SEIF</td>
<td>Sparse Extended Information Filter</td>
</tr>
<tr>
<td>SIR</td>
<td>Sampling Importance Resampling</td>
</tr>
<tr>
<td>SLAM</td>
<td>Simultaneous Localization and Mapping</td>
</tr>
<tr>
<td>SPI</td>
<td>Sterling Polynomial Interpolation</td>
</tr>
<tr>
<td>SS</td>
<td>State Sampling</td>
</tr>
<tr>
<td>TJTF</td>
<td>Thin Junction Trees Filter</td>
</tr>
<tr>
<td>TS</td>
<td>Taylor Series</td>
</tr>
<tr>
<td>TSE</td>
<td>Taylor Series Expansion</td>
</tr>
<tr>
<td>UHS</td>
<td>Unscented HybridSLAM</td>
</tr>
<tr>
<td>UKF</td>
<td>Unscented Kalman Filter</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The Simultaneous Localization and Mapping (SLAM), has been known to be an important problem for navigation purposes for sometimes now. SLAM describes a situation in which an autonomous vehicle builds a map of its surroundings in an unknown environment then uses the momentary map to localize itself. Once the robot is localized, it starts to map the environment from its new location. This process of constructing a map of the environment from a new location and localizing itself based on that particular map occurs simultaneously. Therefore, solving SLAM problem makes the mobile robot independent from a given map a priori. Extensive research on the subject has confirmed that a solution to a real time SLAM problem can be feasible. As real time localization and mapping are considered jointly in SLAM, it may be presented as an estimation problem. In this research, the navigation problem is investigated with respect to a feature point environment representation. As a result, the performance of currently used methods such as EKF-SLAM, FastSLAM, and HybridSLAM are compared and thoroughly discussed.

While Extended Kalman filter (EKF) is well known as the gold standard solution to SLAM problem, it should be noted that it converges only under specific circumstances. The optimality of the filter remains for a high frequency system through which motion and observation error distributions stay uni-modal Gaussian with zero mean. Moreover, the non-linear system has to be linearizable when using EKF. Two major problems of algorithms based on Kalman filters are their computational complexity and their sensitivity to failure in data association. Even though one might argue that FastSLAM, based on Rao-Blackwellised particle filtering (RBPF) suffers from other problems, it can deal with the non-linearity of the
system much better, resulting in the elimination of the linearization step in the SLAM structure. It is important to note that while the FastSLAM algorithm performs well in the vicinity of nearby landmarks, it is sensitive to a relatively large amount of noise variation between motion and observation, specifically, when the observation noise becomes close to zero. This sensitivity happens because statistical samples do not carry enough weight to be incorporated into the calculation. A compromise between both filters would establish by combining FastSLAM and EKF-SLAM, through which, advantages of both filters would be added and shortcomings be eliminated. Essentially, by converting FastSLAM to a Gaussian mixture model and using moment matching technique, the data produced by both filters is interpreted and fused to a final map. The combined filter called HybridSLAM does not have to deal with linearization of the motion. Moreover, HybridSLAM is performing with a high level of robustness in localization and mapping, and at the same time, through a sub-mapping fusion strategy, HybridSLAM method outperforms both previous methods with a high level of estimation accuracy. In HybridSLAM structure however, the elimination of linearization process is limited to the local scale. If non-linearity of the motion is severe, error increases in the particular part of the map that is being built by EKF-SLAM.

This study investigates modification of a fused map with the use of Unscented Kalman Filter (UKF) in a HybridSLAM filter. The proposed algorithm is called Unscented HybridSLAM (UHS) which outperforms the three above algorithms. The advantages and disadvantages of each particular algorithm will be thoroughly discussed in this dissertation where with the help of various simulations, a comparison through performances of different approaches is demonstrated and results are discussed. Furthermore, the efficiency and consistency of the modified method are validated by comparing it to the original HybridSLAM.
1.1 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping [1, 2, 3, 4, 5, 6, 7] describes a situation in which a mobile robot is navigating in an unknown environment without any access to a priori map. Meanwhile, and through its ego-motion, the mobile robot senses the environment to localize itself and at the same time, incrementally, builds a navigational map of its surroundings. Localizing and mapping are highly correlated factors when a solution to SLAM problem is acquired [4]. It is important to note that the initial condition of the robot should be established prior to any attempt to solve the problem. Furthermore, both sensor data and motion of the robot are corrupted by disturbing noise produced by odometric and observation measurements. Figure 1.1 illustrates the SLAM problem schematically. The ellipses indicate the uncertainty of robot path and landmark positions which increase through the process [7]. The uncertainty over landmarks grows with the uncertainty of the path in an unbounded mean. The reason is that, in SLAM problem, there is a high correlation between the robot’s pose and a landmark position. There are elements and sets that describe SLAM as follows [4]:

- A discrete time index \( k = 1, 2, \ldots \)
- \( x^R_k \): Vector of the true location and orientation of the robot at the discrete time \( k \).
- \( u_k \): A known control vector applied at time \( k-1 \) to drive the vehicle from \( x^R_{k-1} \) to \( x^R_k \) at time step \( k \).
- \( m_i \): The vector of true location or parameterization of the \( i^{th} \) landmark.
- \( z_{k,i} \): Observation of the \( i^{th} \) landmark taken from a location \( x^R_k \) at time step \( k \).
- \( z_k \): The generic observation of one or more landmarks taken at time step \( k \).
• The set of history of states or path of the robot:

\[ X^R_k = \{ x^R_0, x^R_1, \ldots, x^R_k \} = \{ X^R_{k-1}, x^R_k \} \]

• The set of history of control inputs:

\[ U_k = \{ u_0, u_1, \ldots, u_k \} = \{ U_{k-1}, u_k \} \]

• The set of all landmarks:

\[ m = \{ m_1, m_2, \ldots, m_M \} \]

• The set of history of observations (Landmark observations):

\[ Z_k = \{ z_1, z_2, \ldots, z_k \} = \{ Z_{k-1}, z_k \} \]

Where \( z_k = \{ z_{k,i}, z_{k,j}, \ldots, z_{k,n} \} \) is a set of all possible landmark observations at time step \( k \).

Also the \( i^{th} \) observed landmark at time step \( k \) is expressed as \( m_{k,i} \) or simply \( m_i \). The reason for skipping sub-script \( k \) is that the landmarks are considered static in this representation and therefore they are time independent. So there will be no need of time step for any non-moving landmark.

### 1.2 SLAM Applications

One of the most practical uses of SLAM is when a mobile robot navigates in an unknown environment. The term unknown simply means that the robot has no access to a map a priori. When a GPS signal is available, receiving data can correct error from odometric measurements or can correct the error generated from natural sources. In many applications, however, there is no such accurate global positioning data available [7]. For instance, when a mobile robot is landing on another planet for exploration purposes or for situations in which the robot travels underground. In such circumstances, the mobile robot is on its own and
needs to carry out an observation device to record its momentary position independent of dead reckoning.

Figure 1.1: A robotic vehicle that moves in an unknown environment and takes observations of many landmarks using a range/bearing sensor [4].

To navigate reliably and to deploy a mobile robot with minimal infrastructure in such environments, the robot needs the help of a technique to combine measurements received from different devices [8]. SLAM has shown its capability of a safe navigation and has come up with promising results so far. On July 7th 2003, NASA launched the “Mars Exploration
Rover” (MER) mission. The mission involved two mobile robots, “Spirit” and “Opportunity” landing on planet Mars to explore the red planet [9]. The mission was followed by landing the third rover “Curiosity” on Mars in March 2012. SLAM was successfully applied in NASA’s different programs, and by far, “Curiosity” rover is still active and sending extremely valuable information to earth.

1.3 Feature-based SLAM

For navigation purposes, there is a need for discrete set of features in the environment that can be easily re-observed by a robot. Early SLAM algorithms represented the environment through a set of simple discrete landmarks described by geometric primitives such as points, lines or circles. Any parameterized model can be selected to represent map of an environment which is commonly assumed to be a collection of point features each described by a number of continuous state variables. As a result, the model indicates collection of points or landmarks in the Global coordinate system [10]. One way to imagine how a robot finds its way in an environment is to picture a person finding his way in a crowded room with closed eyes. As the person reaches and touches every object, he imagines a map and according to that imaginary map, he steps forward and this process goes on concurrently. In a similar way, a robot identifies landmarks in an environment using its range/bearing sensor [11]. In indoor applications, landmarks can be considered doors, walls, other regular household objects and/or artificial signs designed by humans. In outdoor applications, landmarks could be trees, bushes, stones or any artificial landmarks such as studs and signs [12]. Whatever application and environment the landmarks are considered for, they must be easily re-observable by sensors and distinguishable from each other. Landmarks are usually stationary in SLAM
applications but dynamic landmarks have been recently considered for some applications such as tour-guide mobile robots in museums [13].

1.4 Probabilistic SLAM

In SLAM problem, information from odometric and observation sensors (relative and absolute position information) can be added up in form of a unique data [13]. To process the data successfully, there is a need to express it in a mathematical form. In this section, the focus will be on how different position measurements may be combined in a formal probabilistic framework.

1.4.1 Probability Distribution

The general SLAM problem as previously discussed can be described in form of a probability distribution function [14]. If the initial state of a mobile robot $x^R_0$ (state at the time step zero) and the recorded landmark observations $Z_k$ and control inputs $U_k$ up to and including time step $k$ are known, the joint posterior density of the landmark locations and the state of the robot can be recursively described by following probability distribution function [15].

$$P (x^r_k, m | Z_k, U_k, x^r_0)$$ (1.1)
1.4.2 Process Models

If state of motion of the robot, $x^R_k$ and the true locations of landmarks $m$ are known, observation and motion models describe the probability of making an observation of all landmarks $z_k$ and changing of location of a mobile robot simultaneously [16]. Therefore, we can describe the sensing and motion of the robot in a probabilistic form. For simplicity, and throughout this thesis, the motion of the robot and map of landmarks will be noted as $x_k$ in form of the following augmented set and it will be denoted as state of the whole system [10].

$$x_k = \{x^R_k, m\} \quad (1.2)$$

1.4.2.1 Modeling Sensor

The observational model may also be described in probabilistic terms [4, 17]. This probability can be described as a density function considering range sensor observations $z_k$ from a certain location $x_k$ at time step $k$ as follows:

$$P (z_k \mid x_k) \quad (1.3)$$

This probability distribution function is called the sensor model or perceptual model [18]. The transition density is usually difficult to compute and it could be due to the high dimensionality of the measurement. For instance, if a camera set is used for the observation purpose, the probability density becomes very complex caused by the receipt of numerous camera pictures
at each possible location. This setting requires a large amount of computing power and memory for the processing unit [19].

### 1.4.2.2 Modeling Motion

An autonomous robot performs actions in an environment that changes the position of the robot [7]. This action is due to a sequence of control actions. If \( u_k \) from the set of control actions \( U_k = \{u_1, u_2, \ldots, u_k\} \) is the action performed by the robot at time step \( k \), change in the location of the robot can be expressed probabilistically by a transition density function as:

\[
P(x_k \mid x_{k-1}, u_k)
\]  

(1.4)

This density function gives the probability that if at time \( k-1 \) the robot was at location \( x_{k-1} \), and performed a control action \( u_k \), it ends up at location \( x_k \), at time step \( k \). This transaction density is therefore the motion model describing how the control input \( u_k \) changes the location of the robot. It is interesting how actions contain relative information about the new state of the robot in this probability density. Given the last location, the robot can estimate its current location based on the performed control action [20]. Once the state of the robot and the landmark locations are known, it can be assumed that observations are conditionally independent.

### 1.4.3 Bayesian SLAM

SLAM can be described as a Bayesian estimation problem. A Bayesian framework that estimates this density is the Markov Localization framework. This framework captures the
probabilistic foundations of many currently used localization methods and combines information from different sensors in form of relative and absolute position measurements to form the belief of a robot. Estimation of location of the robot and also landmark locations can be described with the consideration of noise in all measurement devices [20].

1.4.4 Belief

With a probabilistic point of view, it can be said that the robot has a belief of what the position of landmarks and its pose. At time step \( k \), the robot is not considering one specific location but a number of locations and landmarks positions. Through the Markov Localization framework, this belief can be described satisfactorily. Markov Localization is an iterative way of using the Bayesian Theorem to update the available information at each time-step where the new sensor reading is available. The Markov Localization framework combines the data from different sensors to form a combined belief regarding the location of a robot and landmarks. The belief can be described by a probability density function over all poses \( x_k \in S \) where \( S \) is the set of all locations of the robot and landmarks (The global coordinate system reference). Therefore, the belief can be described as follows.

\[
Bel(x_k) = P(x_k | X_{k-1}, U_k, x_0)
\]  

(1.5)

The probability distribution in equation (1.5) indicates the belief at state \( x_k^R \) at time step \( k \), given the initial state \( x_0^R \) and all locations of the robot, map of landmarks \( m \), and the set of control actions up to time step \( k \). This probability distribution has the highest possible
probability at which the system can be. The goal of SLAM is to make this belief as close as possible to the real distribution of the system. There is only one peak out of this distribution at the true locations and it is zero everywhere else [15].

1.4.4.1 Prior and Posterior Beliefs

When the robot is navigating through an environment, it incorporates all data coming from different sensors. At time step $k$, the system has a belief of where it is. This belief is called prior belief of the map and its own location. Once the robot incorporates the measurement information from the observation $z_k$ at time step $k$, a new belief is created in the system. This new belief, called the posterior belief, is indicated as $Bel^+(x_k)$. This posterior belief is the belief that the system has, after it has already included the latest observation information into the calculation. With above probabilistic models for sensor and motion, the belief may be updated in a probabilistic term. The initial condition should always be mentioned when the beliefs are updated. The belief of the system after the control action $u_k$ at time step $k-1$ and before incorporating the new observation $z_k$, is called the prior belief and is described as

$$Bel(x_k) = P(x_k | z_1, u_0, z_2, u_1, \ldots, z_{k-1}, u_k, x_0) = P(x_k | Z_{k-1}, U_k, x_0)$$  \hspace{1cm} (1.6)$$

After obtaining observation $z_k$ and incorporating it to obtain the final belief, the posterior belief is described by

$$Bel^+(x_k) = P(x_k | z_1, u_0, z_2, u_1, \ldots, z_{k-1}, u_k, z_k, x_0) = P(x_k | Z_k, U_k, x_0)$$ \hspace{1cm} (1.7)$$

The probability densities of both prior and posterior beliefs can now be computed.
1.4.4.2 Total Probability and Markov Localization

Using total probability and the Markov localization [12, 14, 20], an efficient formula for the next step can be derived as follows:

\[
Bel(x_k) = P(x_k | z_1, u_0, z_2, u_1, \ldots, z_{k-1}, u_k, x_0) = P(x_k | \mathbf{Z}_{k-1}, U_k, x_0)
\]

\[
= \int_{x_{k-1}} P(x_k | x_{k-1}, \mathbf{Z}_{k-1}, U_k, x_0) \times P(x_{k-1} | \mathbf{Z}_{k-1}, U_k, x_0) \, dx_{k-1}
\]

(1.8)

The location of the robot at time step \(k-1\) has nothing to do with the control action \(u_k\). Therefore, the control action \(u_k\) does not have to be taken into account when expressing the prior belief of the robot. Accordingly, and based on the formulation of the posterior belief, equation (1.8) can be rewritten as follows:

\[
Bel(x_k) = \int_{x_{k-1}} P(x_k | x_{k-1}, \mathbf{Z}_{k-1}, U_k, x_0) \times P(x_{k-1} | \mathbf{Z}_{k-1}, U_k, x_0) \, dx_{k-1}
\]

\[
= \int_{x_{k-1}} P(x_k | x_{k-1}, \mathbf{Z}_{k-1}, U_k, x_0) \times Bel(x_{k-1}) \, dx_{k-1}
\]

(1.9)

To further simplify equation (1.9), Markov assumption can be considered. Markov assumption states that based on information of the current state, the previous state of the system become independent of the current state [14]. Therefore, the prior belief can be more simplified. Markov assumption states that it does not matter what the sensor reading was and how the system used the observation information to reach to the new state. In other words, only the current system information is of importance at the current time step. Markov assumption leaves all the previous unnecessary control and observation information behind and will make equation (1.9) much simpler. Therefore, equation (1.9) can be rewritten as
\[ Bel^r(x_k) = \int_s P(x_k | x_{k-1}, U_k, x_0) \times Bel^r(x_{k-1}) \, dx_{k-1} \] (1.10)

The observation data can be incorporated into the prior belief to create the posterior belief defined in equation (1.7). Using Bayesian theorem and the definition of the prior belief, equation (1.7) can be rewritten as

\[ Bel^r(x_k) \times P(z_k | Z_{k-1}, U_k) = P(z_k | x_k, Z_{k-1}, U_k, x_0) \times P(x_k | Z_{k-1}, U_k, x_0) \] (1.11)

Using the Markov assumption, equation (1.11) can be more simplified and rewritten as

\[ Bel^r(x_k) \times P(z_k | Z_{k-1}, U_k) = P(z_k | x_k) \times P(x_k | Z_{k-1}, U_k, x_0) \] (1.12)

or

\[ Bel^r(x_k) \times P(z_k | Z_{k-1}, U_k) = P(z_k | x_k) \times Bel^r(x_k) \] (1.13)

or simply

\[ Bel^r(x_k) = P(x_k | Z_{k-1}, U_k, x_0) \]

or

\[ Bel^r(x_k) = P(x_k | Z_k, U_k, x_0) \]

or simply

\[ P(x_k | Z_k, U_k, x_0) \times P(z_k | Z_{k-1}, U_k) = P(z_k | x_k) \times P(x_k | Z_{k-1}, U_k, x_0) \] (1.14)

Equation (1.14) provides a recursive process to calculate the posterior belief \( Bel^r(x_k) \) for the system state \( x_k \) at time step \( k \) by incorporating the observation data \( Z_k \) and the control actions \( U_k \) up to and including time step \( k \) [3]. This equation is a Bayesian definition of pose of the
robot accompanying a sequential control actions and sensor measurements considering the initial conditions.

### 1.5 Main Contribution of this Thesis

This research presents an original investigation into map integration for navigation using so-called sub-map fusion and Simultaneous Localization and Mapping (SLAM) problem for the first time. The work develops and analyzes a rigorous mathematical solution to integration and/or fusion of a local map produced by Rao-Blackwellised particle filtering into a global map estimated by unscented Kalman filtering SLAM. Map fusion is performed using Constrained Local Sub-Map Filter (CLSF) by allowing FastSLAM mapping update to be scheduled at appropriate intervals. The main contributions of this research is

- Development of Unscented HybridSLAM method based on Bayesian estimation. This method assures a correct correspondence between path of the robot obtained from FastSLAM and a fused map obtained from CLSF at each time step.
- Improvement of theoretical aspects behind SLAM problem and its derivatives, in particular identifying correlations between the motion and observation estimation errors.
- Development of the map fusion method based on sub-map fusion technique resulting in a lower bound frequency error.
- A new approach to the SLAM problem that uses strengths of both Unscented Kalman filter and particle filter to build up a reliable environment map. This approach
reformulates SLAM to reduce the map error issue and thus provides a solution to the SLAM problem, specifically in loop closing scenarios.

- The new approach to the SLAM problem is simulated for an outdoor implementation in different scenarios to verify the theoretical developments and identifying practical considerations.

1.6 Thesis Overview

Chapter 1 of this dissertation deals with different instances of “Simultaneous Localization and Mapping” problem. A probabilistic framework and the formulization of the uncertainty and beliefs in the SLAM problem based on a Bayesian network theory are described in this chapter as well. In chapter 2, the main problem of Simultaneous Localization and Mapping is described to provide the reader concrete examples of how uncertainties could play a role in mobile robots context. Afterwards, a discussion on the prior work in the field is discussed, concentrating primarily on disadvantages of currently used algorithms that are mainly applied to solve SLAM problem.

An improved version of HybridSLAM algorithm called Unscented HybridSLAM producing better results than currently used algorithms is proposed in Chapter 3. It will be demonstrated how Unscented HybridSLAM incorporates the current observation into the new formulation of the problem which results in remarkable computational cost reduction. It will also show how the proposed formulation produces more accurate results while a local map is produced and fused to a global map previously estimated by an unscented Kalman filter. Furthermore, the proof of convergence for FastSLAM proposed by Montemerlo and as a suboptimal filter in Unscented HybridSLAM is represented.
Chapter 4 will further describe the SLAM subject applied on a realistic outdoor car-like robot. First, the Linearization to the first order of Taylor’s series is formulated, followed by using the second order “Sterling Polynomial Interpolation”. Then, the linearization process will be represented and the algorithm will be reformulated to apply different simulated scenarios in chapter 5 along with discussions of details and future work. Chapter 6 contains summary and conclusion.
Chapter 2

Problem Description

This chapter analyzes the structure of SLAM technique. Moreover, the most common algorithms that are currently used to solve SLAM problem will be briefly explained and shortcomings of these algorithms will be investigated and discussed.

2.1 Problem Definition

In the navigation problem, localization is defined as a determination process of the pose of an autonomous mobile robot. Location of a robot in a planar environment is referred to (x, y), the coordinates, and θ, the orientation or heading. In localization problem, a map and the initial location of the robot are known and the goal is to keep track of the position while following a given map. There is always uncertainty in the location of the robot but this problem can be easily solved by applying tracking or local techniques if the robot has access to a solid and reliable map.

On the other hand, the goal of mapping is for a mobile robot to build a map while following a trajectory. The concept of trajectory refers to a situation when the robotic vehicle path $X^R_k$ is known in advance and there is no need for path estimation, resulting in the robot making inferences to acquire a two-dimensional map. To acquire this map, the robot must use its sensor in order to perceive features in the environment. The mapping problem is a straightforward problem since the applied algorithm does not deal with the path estimation.

When estimating path and map together, the uncertainty in the motion affects uncertainty in observation and makes it grow at time step k. At the next time step, k+1, the accumulated
uncertainty of the map of landmarks up to the previous time step, will be propagated to the motion. As the estimation process occurs simultaneously, due to the high correlation between uncertainties in motion and observation processes, the uncertainty of the system grows unbounded and the process of estimation fails. The resulting map will then be a wrong presentation of the environment which leads the robot to a wrong pose at each time step. Figure 2.1 demonstrates a simulation of an SLAM situation in which uncertainties in motion and observation grow as the robot proceeds to complete a loop. The red line indicates the estimated path and the black line connects waypoints. The uncertainty in the mapping of landmarks is in elliptical form in green and the uncertainty in the pose of the robot is the red ellipse. In figure 2.2, black asterisks show the true location of landmarks. The uncertainty is demonstrated as a green ellipse and the final location of the landmark is estimated where the red dot appears.

The probabilistic representation of the system described in equation (1.1) is descriptive enough to model the correlation of uncertainties between motion and observation models. Therefore, it is beneficial to note the generated errors in the probabilistic distribution functions motion and observation models as described in equations (1.3) and (1.4) in the previous chapter. The main goal to solve the SLAM problem (as a parameterized continuous function), is to consider noise which corrupts the system in motion and observation measurements. When the state of a dynamic system is noisy, the state estimation process will be rather difficult. Therefore, there is a need for a filter that can deal with the computational complexity of beliefs over continuous spaces and to estimate the state of this noisy system.
Figure 2.1: Uncertainties in motion and observation grow unbounded during the process.

Figure 2.2: Estimation of a landmark indicated by the red dot.
There are many algorithms implemented as mathematical tools to arrive at a solution based on the Bayesian representation of SLAM problem. Amongst all, EKF-SLAM, FastSLAM, and HybridSLAM (as a combination form of EKF-SLAM and FastSLAM) are of primary interest in this research specially when the uncertainty analysis in the center of the attention.

### 2.2 Uncertainty

Determining the location and simultaneously constructing a map in an unstructured environment is a critical problem in navigating an autonomous mobile robot. In a two-dimensional environment, the location of an autonomous robot can be described by its coordinates \((x,y)\), and its orientation \(\Theta\). In the navigation problem, an autonomous vehicle is meant to build a map of its environment while it is demanded to follow its desired path. There is, however, some error associated with the motion of the robot [11]. For instance, in a four-wheeled mobile robot, the robot controller uses the information provided by the optical encoders attached to the wheels to command the drive system to keep the robot near its desired path. In some cases, however, due to imperfections, the robot drive mechanism may not follow commands very well and as a result the error from encoder readings accumulates and becomes the major source of uncertainty in the system.

Due to the structure of SLAM, uncertainty in the motion model is highly correlated with the uncertainty in observing devices. These uncertainties could grow to a point that it becomes impossible for a robot to make a correct inference of its position. Adding to the encoder error, the slippage due to uneven pathways inaccuracy of range/bearing sensor readings, and the changes of wheel diameter in time can accumulate a significant amount of error in the system. All the above sources of error may result in inaccurate position estimation. Therefore, the
robot needs to have some notion of the error from all different sources. This error consideration can be taken into account when the robot is trying to stay on its path [21]. The error is analyzed from two different points of view; error in motion which causes uncertainty in the location of the robot and error in estimation of landmarks which causes uncertainty in the location of landmarks.

2.2.1 Uncertainty Analysis

Despite of improving accuracy in sensors and measurement devices, uncertainty in motion is unavoidable. Even with the most costly infrastructure and well-engineered systems, the error exists and causes uncertainty in the motion. As the vehicle travels on its path, the uncertainty is accumulated and consequently the system fails. As discussed, it is not possible to eliminate uncertainty but it might be possible to reduce and maintain it within some tolerance. To do so, the error must be reduced as much as possible. The error is simply referred to the misplacement of the vehicle along x and y axes and the unwanted change in the desired angle at every time step. If the error is not maintained at some tolerable limits, the uncertainty in position and orientation of the robot will increase which in turn propagates into the estimation process of landmarks’ locations in the vicinity of the vehicle.

There are many methods to examine the margin of error when the estimation is in process. The traditional method to do so is the numerical computation which examines the maximum error during the process. In this thesis, the error is examined through the measurement of “Root Mean Squared” position error, orientation error, 1-sigma and 2-sigma deviations, and confidence intervals. The goal is to compare the estimated error with the absolute error and to see if it stays within a tolerable margin. In chapter 5 many aspects of the uncertainty analysis
will be shown and through different scenarios, the margin of error in SLAM for different algorithms will be depicted.

### 2.2.2 Uncertainty in Motion

As discussed in section 2.2.1, the error in motion causes unwanted change in orientation and along x and y axes. The main method to scale the margin of error is the “Root Mean Squared” position error which is the sample standard deviation of the differences between estimated position along x and y axes and is represented by

\[
\text{RMS Position Error} = \sqrt{\frac{x^2 + y^2}{2}}
\]  

(2.1)

The deviations are usually the square roots of variances, showing the certainty that the true state lies within a certain distance from the estimated state. In the same way, the orientation error is measured straightly by deducting the estimated and the absolute error as

\[
\text{Orientation Error} = \theta_{\text{estimation}} - \theta_{\text{absolute}}
\]  

(2.2)

Another way to measure the error is to compare Innovation and standard deviation for orientation in 2σ confidence interval which will be an standard method of measuring how much the estimated values lie in the interval of a normal distribution which will be thoroughly discussed in chapter 5.
2.2.3 Uncertainty in Observation

The uncertainty in motion propagates in the observation process which in turn grows the uncertainty in estimation of landmarks locations. Furthermore, the observation devices are not error free and this problem makes them to generate uncertainty in the estimation process. This time the uncertainty propagates through the motion process since the vehicle relies on the map built in the previous time step. Therefore, there is a need to control the error during the observation process which in turn demands some methods to measure and reduce error.

1σ and 2σ confidence intervals are uncertainty measurement tools to show if the error lies between the deviation lines. If the error remains within the 1σ and 2σ uncertainty lines during the estimation process, the error stays within tolerable margins and uncertainty does not grow unbounded. Figure 2.3 depicts a situation where the absolute error is in the margin and under the deviation.

Figure 2.3: Deviation and Absolute error
Measuring the standard deviation of landmarks in the neighborhood of the vehicle would show if their estimated error become equal to the Full SLAM which indicates whether the algorithm is consistent or not. Usually, if the actual error and uncertainty are reduced the map becomes correlated and the filter converges.

### 2.3 Currently Used Algorithms

EKF-SLAM and FastSLAM are two commonly used algorithms for indoor and outdoor applications. However, there are some shortcomings that limit the use of such algorithms in real applications.

#### 2.3.1 EKF-SLAM

EKF-SLAM algorithm is known as a classical solution to the SLAM problem. The EKF algorithm providing an optimal solution for SLAM problem was first introduced by Smith and Cheesman in 1986 [3]. This algorithm is based on linear Kalman filter [22] and it is necessary for the process to be under Gaussian conditions. Furthermore, the system should be linear or at least not much far from it, in order to be able to change to a linear form if needed [23].

Using EKF algorithm, the estimation of the state, and its covariance matrix of the posterior belief $Bel^\ast (x_k) = P(x_k \mid Z_k, U_k, x_0)$, can be computed easily. As illustrated in figure 2.4, the estimation process is completed in three major steps; prediction, observation, and update [24]. Since the EKF algorithm can be implemented as a mathematical tool to achieve a Bayesian solution, the belief of where a robot is, will be represented as a parameterized
continuous function. EKF suffers from two major problems with respect to large scale maps; computational complexity and data association (see appendix A).

![Diagram](image_url)

**Figure 2.4:** The EKF algorithm as a classical solution for SLAM problem when the system is represented with Gaussian implication. $\hat{x}_k$ is the prior state of the robot, $\hat{x}_k^+$ indicates the posterior state of the robot, and $K_k$ is the Kalman gain [6].

### 2.3.1.1 Computational Complexity

One major issue of EKF-SLAM solution is the computational complexity which always exists in the heart of the algorithm. The more landmarks observed and used to build a map, the more time the algorithm consumes. Moreover, EKF algorithm demands a huge memory while the landmarks are added to the map. In fact, the Gaussian representation of the system and noise employed by EKF algorithm makes complexity of the computation quadratic. This is due to the fact that for a two-dimensional environment (which is considerable in most SLAM cases), the system covariance matrix $P$, contains $2M+3$ by $2M+3$ arrays. The number “3” represents
pose variables including (x,y) coordinates, Θ the orientation, and M for the number of landmarks in the map. Therefore a huge memory is required for a quadratic grown covariance matrix as of $M^2$. Leonard and Durrant-Whyte [1] have shown that the EKF-SLAM algorithm is an inappropriate solution for a large scaled environment with a significant amount of landmarks, even though the noise is considered Gaussian. For more details see [2, 4, 25].

Figure 2.5 depicts the system covariance matrix $P$ in the heart of the system. This matrix representation is called the absolute representation of the main matrix of the system, and contains the covariance in the robot position, covariance on the landmarks, covariance between robot’s position and landmarks, and finally covariance between landmarks mutually.

The first cell, $A_{11}$, is the covariance matrix on the robot’s position (x, y, Θ) which is a 3×3 matrix, $A_{22}$ is the covariance matrix on the first landmark which is 2x2 since landmarks are considered static and do not need any orientation. The matrix continues down to $A_{MM}$ which represents the covariance of the last observed landmark. The cell $A_{21}$ contains the covariance between the robot’s state and the first landmark. The cell $A_{12}$ contains the covariance between the first landmark and the robot’s state. $A_{12}$ can be deduced from $A_{21}$ by transposing the sub-matrix $A_{21}$. $A_{M2}$ contains the covariance between the last landmark and the first one, while $A_{2M}$ contains the covariance between the first landmark and the last one, which again can be deduced by transposing $A_{M2}$.

### 2.3.1.2 Single Hypothesis Property

The mapping problem which is solvable by landmarks observations is always influenced by data association. Using the maximum likelihood rule, every observation is assigned to a landmark. Before the data is fused to the map, a new observation of landmark is assigned to it.
If for any specific reason this observation is wrong, which means the probability of the observation is not high enough, a new landmark with no accuracy is added to the map. Since EKF algorithm does not include data association uncertainty, wrong information is added to the map and EKF cannot revise this wrong data. If more inaccurate information is fused to the map at next time steps, the EKF algorithm diverges. Closing the loop will be a major problem specifically when the robot returns to the first landmark after the completion of one loop. In

<table>
<thead>
<tr>
<th>$A_{11}$</th>
<th>$A_{12}$</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$A_{21}$</td>
<td>$A_{22}$</td>
<td>...</td>
<td>...</td>
<td>$A_{2M}$</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$A_{M2}$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>$A_{MM}$</td>
<td>...</td>
</tr>
</tbody>
</table>

Figure 2.5: Absolute representation of covariance matrix of the system $P$. This matrix contains $2M+3$ by $2M+3$ arrays of matrices [6].
figure 2.5 if the number of landmarks increases 5 times, the robot faces a big problem for the loop closing case. For more information on data association see [19].

There have been several attempts to solve the above issues and bring the EKF-SLAM to a better performance but the Gaussian assumption is still a priority. Montemerlo [8] has proposed an alternative algorithm based on Monte Carlo Localization (MCL) to deal with the SLAM problem by implementation of Rao-Blackwellised particle filter (RBPF). This Alternative solution called FastSLAM contains the same promising results of EKF-SLAM with non-Gaussian implication of the system. In section 2.3.2, advantages and disadvantages of FastSLAM algorithm will be discussed in detail.

2.3.1.3 Noise Characteristics

There is always some noise around the sensing devices corrupting the navigation process. If EKF is used to estimate the robot path and locations of landmarks in the environment, then the corrupting noise must carry some specific characteristics. EKF can handle a type of noise which is independent, white and zero mean. Furthermore, this noise needs to be represented in Gaussian form to make the best estimator out of the EKF-SLAM algorithm. Motion and Observation models ought to have a Gaussian representation as well [26]. If all above conditions are not satisfied, the filter is not able to estimate the state of the system properly.

2.3.1.4 System Nonlinearity

Most dynamic systems in the real world applications are non-linear. As previously discussed, Kalman Filters cannot deal with any system non-linearity. Any other extension of KF algorithm is not able to handle motion and sensor models that cannot be linearized.
2.3.2 SLAM based on Rao-Blackwellised Particle Filter

SLAM can be solved according to Rao-Blackwellised particle filtering [27, 28] called FastSLAM [8] which is based on straightforward factorization. This factorization is based on the observation that if the true path of a robot is known, the individual landmarks are mutually independent. In other words, if there is no uncertainty about the robot pose and the observations are independent, the estimation of each landmark in the map is independent of the rest of the map. In reality, the path is unknown in advance. However, the conditional independence enables the filter to estimate the general probability distribution in factored term according to equation (1.1).

\[
P(X_k^R, m | Z_k, U_k, x_0^R) = P(X_k^R | Z_k, U_k, x_0^R) P(m | X_k^R, Z_k, U_k, x_0^R)
\]

\[
= P(X_k^R | Z_k, U_k, x_0^R) \times \prod_{i=1}^{M} P(m_i | X_k^R, Z_k, U_k, x_0^R)
\]

(2.3)

The first factor is the estimation of the robot path and the second factor in the above equation is the product of \(M\) landmarks estimation given the robot pose is known. This factorization is the fundamental idea behind FastSLAM method. The algorithm decomposes the SLAM problem into a localization and \(M\) landmark position estimation problems. Furthermore, FastSLAM relies on a so called particle filter to estimate the following robot posterior:

\[
P(X_k^R | Z_k, U_k, x_0^R)
\]

(2.4)
FastSLAM can be updated in constant time for each particle in the filter. To update the map, the algorithm relies on $M$ independent Kalman filters for the $M$ landmark estimates.

$$P(m | \mathbf{X}^R_k, Z_k, U_k, x^R_0)$$ (2.5)

As will be discussed in chapter 3, the entire filter can be updated using logarithmic scale in the number of landmarks $M$ as shown in [8]. FastSLAM method can also handle non-linear robot motion models. Besides this non-linearity of motion advantage, the noise does not have to be Gaussian anymore but it can be any type of noise. For more details see appendix B.

### 2.3.2.1 Recursive Monte Carlo Sampling

FastSLAM with its basis in Recursive Monte Carlo Sampling, directly, represents a non-linear process model which is not necessarily under Gaussian conditions. This method represents distributions using a finite set of sample states or particles. Montemerlo [8] used this approach on the basis of earlier work of Thrun and colleagues [29]. Particle filter is a mathematical tool to overcome discussed issues. Early applications of particle filter in the SLAM problem go back to the work of Thrun, Fox, and Burgard [30] as Monte Carlo Localization, where a set of particles were used to represent the distribution of possible states of a robot relative to a fixed map. Whereby a joint state is partitioned according to the product rule $P(x_1, x_2) = P(x_2 | x_1)P(x_1)$. And if $P(x_2 | x_1)$ can be represented analytically, only $P(x_1)$ needs to be sampled. As shown in figure 2.4, the map is represented as a set of independent Gaussians, with a linear complexity [33], rather than a joint map covariance with quadratic complexity offered in [31].
2.3.2.2 Sample Impoverishment

In FastSLAM algorithm, the proposal and target distributions are matched to draw samples for path estimation (see appendix B). The better these distribution functions fit, the more samples are incorporated to the estimation process, and if for any reason, target and proposal distributions do not match, many particles will be ignored in the re-sampling step.

If the motion is noisy and the sensor is highly accurate, which means the amount of noise in sensor readings is near to zero, some particles are thrown away in the third step of the algorithm. This effect will be demonstrated based on the simulation data in chapter 6. When the measurement noise approaches zero, the proposal and target distributions are mismatched since lots of particles (statistical samples) are thrown away. As a result, there will not be enough samples to estimate the state of the robot at each time step, and consequently, the filter diverges. This process of elimination is called sample impoverishment and is a major disadvantage of FastSLAM algorithm [32].

2.3.2.3 Overconfidence

Since FastSLAM algorithm is not capable of maintaining particle diversity over a long period of time, it operates in a very high-dimensional space of robot’s path [33]. If a small number of particles is used, the uncertainty along the path will be ignored and the estimation process fails. As a result, there is a need for a large number of particles for a relatively long path. For instance, in a loop closing case, full uncertainty is required when the loop is completed. For remembering uncertainty in a long trajectory, any particle-based filter, which is using a small number of statistical samples, fails to estimate correctly and if a large number of particles are
used, a huge processing power is required by the filter. Recent developments of FastSLAM including Unscented FastSLAM [34] suffer from the same issue.

2.3.3 HybridSLAM

EKF-SLAM and FastSLAM have some shortcomings and are limited to certain applications. HybridSLAM (HS) combines the strengths of both EKF-SLAM and FastSLAM. Meanwhile, HS avoids current methods’ weaknesses resulting in a modified algorithm which outperforms both filters. Due to its high level of robustness to clutter and ambiguous data association, FastSLAM is a preferred algorithm for many applications [35]. It is nonetheless, limited to short trajectory applications or cases where the observation noise and motion noise have a relatively large difference [36]. To overcome these problems, FastSLAM should be modified in such a way to decrease its weaknesses and to acquire advantages of EKF-SLAM. One advantage of EKF-SLAM is that it remembers uncertainty over a long path avoiding the tendency of FastSLAM to become over-confident. By combining EKF-SLAM and FastSLAM, the modified filter avoids a substantial estimation error. This modification was done by Brooks and Bailey [37] by presenting FastSLAM as a continuous (Gaussian) form. Incidentally, the optimality of EKF originates from its Gaussian representation.

2.3.3.1 Modification

“The modification may be done through FastSLAM building local maps allowing it to run for long enough to disambiguate associations” [37]. Before the path becomes so long that particle diversity turns problematic, a single dimensional Gaussian is computed from the FastSLAM
posterior. This Gaussian local map can be fused into the whole map” [37]. Prior to this fusion, a decision will be made about the associations between local map features and the whole map features [38]. If the local map can provide sufficient constraints to make the probability of a decision low, the result can be an algorithm which is “robust to linearization errors and data association ambiguities in a local map” [37]. EKF estimates the posterior state of the system according to its probability distribution function, meaning that the estimation is done sequentially over landmark location and the pose at every time step $k$ [39]. In FastSLAM algorithm, the posterior is estimated over the landmark location as well, but instead of the single state of the robot at time step $k$, the path is estimated according to the probability distribution function conditioned on the path of the robot [40]. One major contribution of such distribution function is to produce a whole map of landmarks based on the robot path. If this distribution can be extracted from FastSLAM algorithm and expressed in form of a single multi-dimensional Gaussian form, it is possible to fuse the map into a global EKF-SLAM map. FastSLAM equation indicates that with the notification of robot’s path, a landmark position is conditional to path of the robot and independent of the other landmarks [37]. FastSLAM may be used to produce small pieces of map (local map) and can be fused to the main map produced by EKF-SLAM (global map). As a result, the distribution of this local map must be interpreted to match the global map representation. The global map produced by EKF-SLAM is a single multi-dimensional Gaussian form [37].

### 2.3.3.2 Diversity in Sample Set

HybridSLAM remembers long term uncertainty over the path since main covariance matrix of the system in EKF maintains covariance between the pose and landmarks positions for the
rest of the calculation [41]. Combining EKF-SLAM and FastSLAM, and representing it as HybridSLAM allows the filter to overcome some issues at which either individual filter would be inconsistent. HybridSLAM is taking advantage of FastSLAM in order to deal with non-linearity of the system, estimating the path and mapping in local scale, and being less fragile to the data association problem. On the other hand, EKF-SLAM has the ability to recall uncertainty at each time step when needed. Since EKF-SLAM is able to deal with nonlinearity of the system to some extent, it may be the right choice for some applications [37]. It should be noted that EKF has its own estimation limitations and approximation issues when facing a highly non-linear system. For instance, if a non-holonomic platform with a high level of non-linearity is operated in a field, the EKF part in HybridSLAM may not be able to represent a correct version of the global map and in the fusion step, the resulting map becomes inconsistent, and it certainly does so in a loop closing scenario.

Another difficulty that HybridSLAM is facing would be the sample impoverishment occurring in its suboptimal filter RBPF. The same problem could happen to HS if there is a great difference between the observation noise and motion noise leading to diverse particle sets. Similar to FastSLAM algorithm, HS suffers from mismatching target and proposal distributions and leads the estimation of local map to an incorrect filtering process.

### 2.4 Comparison of UHS with Currently Used Methods

The remainder of this thesis reformulates the SLAM problem by replacing UKF with EKF in HS and a thorough investigation on the performance of the new filter by comparing it to currently used ones. EKF approximate the state distribution using a Gaussian random variable and propagates it analytically through the first order linearization of a non-linear system [42,
Therefore, EKF optimization may lead to suboptimal performance resulting in the filter due to its generation of substantial error in the posterior mean and system covariance [44]. UKF represents the state distribution of the system based on a Gaussian random variable as well and a deterministic sample approach. In UKF based approaches, however, a minimal set of sample points are chosen to calculate the mean and covariance of the Gaussian random variable. The samples are propagated through the high non-linearity of the system and, more accurately, approximate posterior belief of the robot to the second order Taylor series expansion. The approximation using UKF is more accurate compared to EKF which employs only the first order of Taylor series expansion (TSE), specifically when motion and observation models are highly non-linear. In UKF, sigma points (minimal set of sample points) around the mean are used in a deterministic sampling technique called unscented Transform. By propagating sigma points through non-linear functions of motion and observation models, and as a result, a high estimation of true mean and system covariance can be achieved [45, 46]. It is important to mention that unscented transform eliminates Jacobians calculation in UKF by which the complexity of computation is reduced substantially. However, the estimation is done in a prediction and update manner similar to EKF [47].
2.5 Summary

Simultaneous Localization and Mapping is a technique used by autonomous mobile robots to construct a map of an unknown environment while keeping the robot on its desired path. Observation and motion in a dynamic system are both corrupted by noise leading the system to have uncertainty. The uncertainty in motion is highly correlated with the uncertainty in observations, which results the uncertainty of the system to grow unbounded. This problem can be remedied using mean square or statistical techniques mostly based on Kalman algorithm but each technique has its own limitation in real applications. The gold standard technique is the EKF-SLAM as an extension form of linear Kalman filter, specifically developed for non-linear applications. Nevertheless, the filter suffers from two major problems; computational complexity and its sensitivity to data association. FastSLAM, as an alternative solution, carries a multi hypothesis data association property which is relatively capable of compiling an accurate map of the environment. FastSLAM calculates the map and location in a linear computational form, and as a result, is less costly compared to EKF-SLAM. Nonetheless, for the purpose of very large maps or loop closing cases, the filter is not able to remember uncertainty in the system and becomes overconfident. HybridSLAM is a modified filter based on both Rao-Blackwellised particle filter and EKF-SLAM which takes advantage of the strengths of both filters and overcomes their performances. The HybridSLAM, however, can only deal with a situation in which there is a nominal difference between observation noise and motion noise. In cases where the difference between the two sources of noise is substantial, sample impoverishment occurs and many particles will be eliminated for the rest of estimation process. As a result, a large number of particles is needed and consequently a huge amount of processing time and power is demanded. The goal is then
to acquire an algorithm which outperforms HybridSLAM as well as the other two major alternatives, EKF-SLAM and FastSLAM and comparing of the proposed filter to currently used algorithms to evaluate its performance and validity.
Chapter 3

Unscented HybridSLAM

This chapter discusses different algorithm constructing the main structure of the proposed algorithm; Unscented HybridSLAM. Suboptimal filters in particular Unscented Kalman and FastSLAM filters are represented in details in appendices A and B. Same as EKF, UKF is based on recursive data processing and it has been successfully applied in some applications [48]. The EKF estimates the state of a noisy system using noisy measurements. More precisely, it calculates the conditional probability distribution over the space of states given measurements [49]. This is accomplished in a prediction-correction manner, where it first predicts the state of the system based on system dynamics and then, corrects its prediction based on measurements [50]. Both filters make some assumptions on the system, measurements, and different noise involved in the estimation problem. Kalman filters assume that the system and measurements are adequately modeled by a linearized non-linear dynamic system with independent, white, Gaussian, and zero mean noise. To deal with the complexity in the sensor and action models, the models must be assumed Gaussian distributed. Moreover, Kalman based filters assume that the initial state of the system is also independent and Gaussian distributed.

EKF for small scale environment and Gaussian conditions is an appropriate algorithm for SLAM scenarios. Nonetheless, in case of outdoor applications where number of landmarks becomes extremely large, EKF suffers from two major problems; computation complexity and single hypothesis data association [51]. These two issues make a huge use of memory and computational power and are costly while in case of loop closing the filter diverges [52].
On the other hand, FastSLAM is a method based on Rao-Blackwellised Particle filtering. Since landmarks are mutually independent, observation of each landmark can be incorporated to the map with no correlation with other landmarks in the map. The multi hypothesis data association property makes FastSLAM a straight forward algorithm to reduce ambiguity in observation of landmarks, resulting in a low-dimensional problem. The structure of FastSLAM lets the path of robot be estimated by particles while there is EKF filters used for the number of landmarks in the map. The landmark position will be estimated by EKFs according to the path of the robot estimated by particles in advance.

RBPF in FastSLAM fulfills the task in four major steps; state sampling, landmark estimation update, importance weight, and updating importance sampling. This process is based on drawing samples from the proposal distribution that is being matched with a target distribution. Samples have higher weights in the regions where the proposal distribution is larger than the target distribution. Drawing samples from the proposal distribution simplifies the process rather than drawing samples from the target distribution. Computational complexity is not at the level of concern with regards to EKF-SLAM any more since it is not quadratic. Furthermore, it appears to be linear and even simpler with the logarithmic representation of FastSLAM. With the structure of FastSLAM algorithm, the data association is not a major concern since it possesses multiple hypothesis data association property. FastSLAM however, suffers from one major problem. As the path becomes larger, the filter’s over confidence grows higher to the extent that it diverges and fails to accurately estimate the map and the location of the robot. The filter is also sensitive to a high difference between motion and observation noise. As a result, FastSLAM may not be a proper algorithm for every environment [53].
3.1 Estimation of the Joint Posterior

Discussed algorithms in appendices A and B have some shortcomings and are limited to certain applications. The main goal of this chapter is to combine the strengths of such algorithms and at the same time, avoid their weaknesses to obtain a modified filter. Due to its high level of robustness to cluttered environments, FastSLAM is the preferred algorithm for situations where the ambiguity of data association is involved [54]. It is, nonetheless, limited to applications in which the observation sensor is noisy. To overcome these problems, the filter may be constructed of suboptimal filters in order to conceal the weaknesses and at the same time, use the suboptimal filter advantages to the full extent. One advantage of Kalman-based filters is that they have the ability for uncertainty to be remembered over a long path while avoiding the FastSLAM filter to become over-confident as discussed in [37].

A modification to FastSLAM halts it from encountering difficulties. This modification may be possible by representing FastSLAM with a continuous form (Gaussian) since the optimality of EKF originates from its Gaussian representation. The modification may be done same as [37] for that FastSLAM to build local maps and be allowed to run for long enough to disambiguate data associations. Before the path becomes so long that particle diversity is problematic, a single Gaussian is computed from the FastSLAM posterior. This Gaussian local map can be fused into a global map. Before the local map produced by FastSLAM is fused to the global map, a decision will be made about the associations between local map features and the whole map features [37]. If the local map can provide sufficient constraints to lower the probability of a bad decision, it results in an algorithm with robustness to linearization errors and data association ambiguities. The constraint used to fuse the local map is the Constrained Local Sub-map Filter (CLSF) [40]. This constraint will be further discussed in section in this
chapter. UKF and EKF both estimate the posterior of the state, meaning that the estimation is accomplished sequentially over landmark location and the pose at every time step $k$. In FastSLAM algorithm, the posterior is estimated over the landmark location as well, but instead of the single state of the robot at time step $k$, the path is estimated, where probabilistically, FastSLAM is employing below equation [55].

$$P(\mathbf{X}_k | \mathbf{Z}_k, \mathbf{U}_k, \mathbf{x}_0^R, \mathbf{d}_k)$$

(3.1)

### 3.1.1 Factorization

The difference between Kalman-based filters and particle-based filters is that in Kalman filters, only a single state $\mathbf{x}_k^R$ is estimated as the posterior joint, while in particle filters as defined in equation (3.1), the whole path of the robot $\mathbf{X}_k^R$ up to time step $k$ is computed. One major contribution of equation (3.1) is to produce a whole map of landmarks based on path of the robot [8]. If this distribution can be extracted from FastSLAM algorithm and expressed in form of a single multi-dimensional Gaussian, it is possible to fuse this map into a global map previously estimated by UKF. FastSLAM can be expressed as

$$P(\mathbf{X}_k^R, \mathbf{m} | \mathbf{Z}_k, \mathbf{U}_k, \mathbf{x}_0^R, \mathbf{d}_k) =$$

$$P(\mathbf{X}_k^R | \mathbf{Z}_k, \mathbf{U}_k, \mathbf{x}_0^R, \mathbf{d}_k) \times \prod_{i=1}^{M} P(\mathbf{m}_i | \mathbf{X}_k^R, \mathbf{Z}_k, \mathbf{U}_k, \mathbf{x}_0^R, \mathbf{d}_k)$$

(3.2)
Equation (3.2) indicates that with the notification of the path, a landmark position is conditional to the path and independent of the other landmarks. Consequently, there will be \( M+1 \) filters; one particle filter for the path and \( M \) Extended Kalman Filters for landmarks positions estimation conditioned to the path. When the filter estimates path of the robot using particles, every landmark position is being tracked by EKF. Number of EKFs depends on the number of particles \( P \). Therefore, there will be \( M \times P \) EKFs in total to estimate landmarks locations using path of the robot at each time step. Figure 3.1 illustrates how every particle participates individually in the algorithm to estimate the path as well as carrying properties of each individual landmark at time step \( k \), through which, RBPF may be defined.

\[
\begin{array}{cccc}
\text{State of robot} & \text{Landmark 1} & \cdots & \text{Landmark M} \\
\hline
\text{Particle 1} & x_k^R & \mu_1, \sigma_1 & \cdots & \mu_M, \sigma_M \\
\text{Particle 2} & x_k^R & \mu_1, \sigma_1 & \cdots & \mu_M, \sigma_M \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
\text{Particle P} & x_k^R & \mu_1, \sigma_1 & \cdots & \mu_M, \sigma_M \\
\end{array}
\]

*Figure 3.1: States of the robot in regards to "P" particles while there are M landmarks estimated by \( M \times P \) EKFs [8].*

### 3.1.2 Derivation

Posterior of the robot position explained in FastSLAM is considered to show that for all non-negative values of \( P (m \mid X_k^R, Z_k, U_k, x_0^R, d_k) = \prod_{i=1}^{M} P (m_i \mid X_k^R, Z_k, U_k, x_0^R, d_k) \), one
can obtain the factorization [8]. If landmark \( m_n \) is observed at time step \( k \), the following probability equality is obtained.

\[
P(m_n \mid X^R_k, Z_k, U_k, x^R_0, d_k) P(z_k \mid X^R_k, Z_{k-1}, U_k, x^R_0, d_k) = \]

\[
P(z_k \mid m_n, X^R_k, Z_{k-1}, U_k, x^R_0, d_k) P(m_n \mid X^R_{k-1}, U_{k-1}, x^R_0, d_{k-1})
\]

(3.3)

The observation \( z_k \) depends on the robot state at time step \( k \), and the observed landmark. Thus the current state and control vectors, and also the current data association regarding the landmark observation are not involved in the probability terms of equation (3.3). Therefore:

\[
P(m_n \mid X^R_k, Z_k, U_k, x^R_0, d_k) P(z_k \mid m_n, X^R_k, Z_{k-1}, U_k, x^R_0, d_k) = \]

\[
P(z_k \mid m_i, X^R_k, x^R_0, d_{k,n}) P(m_i \mid X^R_{k-1}, Z_{k-1}, U_{k-1}, x^R_0, d_{k-1})
\]

(3.4)

The equation is re-arranged as follows

\[
P(m_n \mid X^R_{k-1}, Z_{k-1}, U_{k-1}, x^R_0, d_{k-1}) =
\]

\[
\frac{P(z_k \mid X^R_k, Z_{k-1}, U_k, x^R_0, d_k)}{P(z_k \mid m_i, x^R_0, d_{k,n})} P(m_n \mid X^R_k, Z_k, U_k, x^R_0, d_k)
\]

(3.5)

During the navigation, if a landmark is not observed, according to Markov process, the prior and posterior observations of that landmark are registered as same, therefore:

\[
P(m_n \text{ not observed} \mid X^R_k, Z_k, U_k, x^R_0, d_k) = P(m_n \text{ not observed} \mid X^R_{k-1}, Z_{k-1}, U_{k-1}, x^R_0, d_{k-1})
\]

(3.6)
assuming

\[ P( \mathbf{m} | \mathbf{X}_{k-1}^k, \mathbf{Z}_{k-1}, \mathbf{U}_k, \mathbf{x}_0^k, \mathbf{d}_k^k ) = \prod_{i=1}^{M} P( \mathbf{m}_i | \mathbf{X}_{k-1}^k, \mathbf{Z}_{k-1}, \mathbf{U}_k, \mathbf{x}_0^k, \mathbf{d}_k^k ) \] (3.7)

For the zero time step, (when the robot starts moving), the factorization is trivially correct.

For time steps \( k>0 \), and using Bayes rule, one can write

\[ P( \mathbf{m} | \mathbf{X}_k^k, \mathbf{Z}_k, \mathbf{U}_k, \mathbf{x}_0^k, \mathbf{d}_k^k ) = \]

\[ \frac{P( \mathbf{z}_k | \mathbf{m}, \mathbf{X}_k^k, \mathbf{Z}_{k-1}, \mathbf{U}_k, \mathbf{x}_0^k, \mathbf{d}_k^k )}{P( \mathbf{z}_k | \mathbf{X}_k^k, \mathbf{Z}_{k-1}, \mathbf{U}_k, \mathbf{x}_0^k, \mathbf{d}_k^k )} P( \mathbf{m} | \mathbf{X}_k^k, \mathbf{Z}_{k-1}, \mathbf{U}_k, \mathbf{x}_0^k, \mathbf{d}_k^k ) \] (3.8)

According to Markov process, the observation \( \mathbf{z}_k \) depends only on the map \( \mathbf{m} \), state \( \mathbf{x}_k^k \), and the data association \( \mathbf{d}_{k,i} \). Furthermore, position of the landmark does not depend on path state \( \mathbf{x}_k^k \), control \( \mathbf{u}_k \), or even the data association \( \mathbf{d}_{k,i} \). Equation (3.8) can then be re-written as

\[ P( \mathbf{m} | \mathbf{X}_k^k, \mathbf{Z}_k, \mathbf{U}_k, \mathbf{x}_0^k, \mathbf{d}_k^k ) = \]

\[ \frac{P( \mathbf{z}_k | \mathbf{m}_n, \mathbf{X}_k^k, \mathbf{x}_0^k, \mathbf{d}_{k,i} )}{P( \mathbf{z}_k | \mathbf{X}_k^k, \mathbf{Z}_{k-1}, \mathbf{U}_k, \mathbf{x}_0^k, \mathbf{d}_k^k )} P( \mathbf{m} | \mathbf{X}_{k-1}^k, \mathbf{Z}_{k-1}, \mathbf{U}_{k-1}, \mathbf{x}_0^k, \mathbf{d}_{k-1}^k ) \] (3.9)

According to (4.9) equation (4.7) can be re-written as
\[ P (m | X^R_k, Z_k, U_k, x^R_0, d_k) = \]
\[ \frac{P (z_k | m_n, X^R_k, x^R_0, d_k) \prod_{i=1}^{M} P (m_i | X^R_{k-1}, Z_{k-1}, U_{k-1}, x^R_0, d_{k-1})}{P (z_k | X^R_k, Z_{k-1}, U_k, x^R_0, d_k)} \]  

(3.10)

From (equations (3.5) and (3.6) one may have

\[ P (m | X^R_k, Z_k, U_k, x^R_0, d_k) = \]
\[ P (m_n | X^R_k, Z_k, U_k, x^R_0, d_k) \prod_{i=1}^{M} P (m_i | X^R_{k-1}, Z_{k-1}, U_{k-1}, x^R_0, d_{k-1}) \]  

(3.11)

That is equal to the product of the individual landmark posteriors

\[ P (m | X^R_k, Z_k, U_k, x^R_0, d_k) = \prod_{i=1}^{M} P (m_i | X^R_{k-1}, Z_{k-1}, U_{k-1}, x^R_0, d_{k-1}) \]  

(3.12)

### 3.2 Single Gaussian Probability Function

RBPF is used to produce momentary maps (local map) and be fused to the main map produced by UKF-SLAM (global map). The distribution of this local map however, must be interpreted to the global map format. The global map produced by UKF-SLAM is a single multi-dimensional Gaussian. A distribution of multi dimensional Gaussian form can be extracted from RBPF. According to equation (3.1) state of the system at time step \( k \) can be expressed as

\[ x_k^n = \{ X^R_k, \mu_{k,1}, \sigma_{k,1}, \ldots, \mu_{k,M}, \sigma_{k,M} \} \]  

(3.13)
and the weighted sample for particle n at the same time step k is:

\[ {\hat{w}}_k^n = P(Z_k | X_k^n, Z_{k-1}, U_k, x_0^R) \]  \hspace{1cm} (3.14)

Using the two above equations, a new presentation of state of the system can be extracted as

\[ x_k^n = \{ {\hat{w}}_k^n, X_k^n, \mu_{k,1}, \ldots, \mu_{k,M}, \sigma_{k,1}, \ldots, \sigma_{k,M} \} \]  \hspace{1cm} (3.15)

or

\[ x_k^n = \{ {\hat{w}}_k^n, (x_k^R, m), C_k^n \} \]  \hspace{1cm} (3.16)

Where

\[ C_k^n = \begin{bmatrix} Q_k & 0 & \ldots & 0 \\ 0 & \sigma_{k,1} & \ldots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \sigma_{k,M} \end{bmatrix} \]  \hspace{1cm} (3.17)

Equation (3.16) represents RBPF as a Gaussian mixture model in which each particle is a Gaussian component with weight \( {\hat{w}}_k^n \), mean \( (x_k^R, m) \), and covariance matrix \( C_k^n \).

According to [44], with the use of moment matching process, the final result for the pose with covariance \( C_k \) at time step k extracted from all particles can be expressed as

\[ x_k = \sum_n {\hat{w}}_k^n x_k^n \]  \hspace{1cm} (3.18)

\[ C_k = \sum_n {\hat{w}}_k^n [C_k^n + (x_k^n - x_k)(x_k^n - x_k)^T] \]  \hspace{1cm} (3.19)
Since the observation sensor (range/bearing sensor) is not noiseless, $^nC_k$ represents the map covariance produced by every individual particle. As a result, a noisy motion $(^n\mathbf{x}_k - \mathbf{x}_k)(^n\mathbf{x}_k - \mathbf{x}_k)^T$ represents the variation between the map produced by particles. At each time step, a new local map is developed by RBPF and fused into global map (the map produced by UKF-SLAM).

### 3.3 Function Linearization

If the model is already in form of a linear function, outputs will be predictable in time over iterations. Since the linear theory is complete and fully developed, differential equations are easy to solve and the computation will not have any difficulties at all. To estimate the posterior state of a non-linear dynamic system, the system must be linearized. Linearization around a point simply means that the function is approximated at a very small vicinity of that point. Linearization techniques simulate linear behavior locally at a point or along a small interval. The results of this simulation are then extrapolated to a general domain where the extrapolation depends on the direction of linearity (the direction of derivatives at a point) [73].

#### 3.3.1 Perturbation

The true values of random variables in a non-linear dynamic system do not follow the nominal trajectory. There is always a small difference between the nominal and true trajectory due to the presence of random noise, such as system, measurement, and initial state noise. These small differences are called perturbations as the main source of uncertainty in the system.
3.3.2 Linearization in UHS

To linearize a nonlinear motion and related observation functions, a nominal trajectory is defined to linearize perturbations that occur around nominal trajectory using the second order Taylor series approximation. Through the Taylor series, the nonlinear functions can be decomposed into two components; a known nominal component and an unknown perturbation component. If the function is infinitely continuously differentiable many times, the influence of perturbations on the trajectory may be then represented by a Taylor series expansion about the nominal trajectory. The Taylor series expansion of a function \( f(x + \Delta x) \) around a point \( x \) is defined as

\[
f(x + \Delta x) = f(x) + \frac{f'(x)}{2!} \Delta x^2 + \frac{f''(x)}{3!} \Delta x^3 + \cdots + \frac{f^n(x)}{n!} \Delta x^n + \cdots
\]

(3.20)

where \( f^n(x) \) is the \( n \)th derivative of function \( f(x) \) with respect to \( x \), evaluated at the linearization point \( x \), and \( \Delta x \) is the perturbation. To linearize a nonlinear function, the Taylor series expansion terms with derivative order higher than two can simply be dropped. There are some points that require attention when linearizing a function in such an order. If the first derivative of the nonlinear function is infinitely large or undefined at the point of linearization, then one cannot linearize at that point. Moreover, the perturbations have to be relatively small compared to the higher-order terms in order to result in a meaningful linearization. The size of the perturbations from the variances of the involved variables can be determined. If the variances are small, proper approximations by ignoring the higher order terms can be obtained. If the variances are not small, the linearization into the second order of the Taylor series may be done through the help of unscented transformation. The motion linearization and observation models will be thoroughly discussed in chapter 5 for an actual
outdoor mobile robot using EKF augmented vector, followed by the unscented transformation for the second order of the system function.

### 3.3.3 UKF Algorithm

To calculate the statistics of a random variable, the unscented transformation method is used below [62]. The system can be expressed as an augmented form of estimated state and system covariance as

\[
x_{k-1}^{(\text{aug})} = [(\hat{x}_{k-1}^+)^T E[ w_k^T]]^T
\]

\[
P_{k-1}^{(\text{aug})} = \begin{bmatrix} P_{k-1}^+ & 0 \\ 0 & Q_k \end{bmatrix}
\]

If \( \kappa \) is defined as the second scaling parameter, and \( \alpha \) is defined as a constant determining the spread of sigma points around the augmented state of the system, \( \lambda \) is defined as \( \lambda = \alpha^2(L + \kappa) - L \) where the constant \( \kappa \) is usually set to 0 or 3\( -L \) and \( \alpha \) is a small positive value \( 0 \leq \alpha \leq e^{-4} \) [63].

The dimension of the augmented state of the system is \( L \), and \( \sqrt{(L + \lambda)}P_{k-1}^+(\text{aug}) \) is the \( i^{th} \) column of the matrix square root of \((L + \lambda)P_{k-1}^+(\text{aug}) \). Since the square root \( A \) of matrix \( B \) satisfies \( B = A.A^T \), 2L+1, sigma points can be derived as

\[
\chi_{k-1}^0 = x_{k-1}^{(\text{aug})}
\]

\[
\chi_{k-1}^i = x_{k-1}^{(\text{aug})} + \left( \sqrt{(L + \lambda)}P_{k-1}^-(\text{aug}) \right)_i
\]
\[ x'_{k-1} = x_{k-1}(\text{aug}) - \left( \sqrt{(L + \lambda) P_{k-1}(\text{aug})} \right)_{k-1}. \] (3.25)

Through the motion function \( f : \mathbb{R}^k \rightarrow \mathbb{R}^{1 \times 1} \), sigma points are propagated as

\[ x'_{i} = f \left( x'_{k-1} \right) \quad i = 0...2L \] (3.26)

Assuming \( \lambda = \alpha^2 (L + \kappa) - L \), and \( \beta \) is a factor to incorporate the Gaussian distribution of the system, factor weights for the state of the system and covariance can be described as

\[ w_{0 \text{sys}} = \frac{\lambda}{L + \lambda} \] (3.27)

\[ w_{i \text{sys}} = w_{i \text{cov}} = \frac{1}{2(L + \lambda)} \] (3.28)

\[ w_{0 \text{cov}} = \frac{\lambda}{L + \lambda} + (1 - \alpha^2 + \beta) \] (3.29)

Recombining the above weighted sigma points, results in predicted state and covariance as

\[ \hat{x}_k^- = \sum_{i=0}^{2L} w_{i \text{sys}} x'_{i} \] (3.30)

\[ P_k^- = \sum_{i=0}^{2L} w_{i \text{cov}} \left[ x'_{i} - \hat{x}_k^- \right] \left[ x'_{i} - \hat{x}_k^- \right]^T \] (3.31)

Mean and covariance of the observation noise are depicted as an augmented vector as

\[ x_k (\text{aug}) = \left[ (\hat{x}_k^+)^T \ E[ v_k^T] \right]^T \] (3.32)
The sigma points in this step are projected through the observation function "h" as

\[
\gamma_k^i = h(\chi_k^i) \quad i = 0...2L
\]  

(3.39)

The weighted sigma points are recombined to produce the predicted observation and predicted observation covariance as

\[
\hat{z}_k^- = \sum_{i=0}^{2L} w_{sys}^i \gamma_k^i \\
\mathbf{P}_{z_k}^+ = \sum_{i=0}^{2L} w_{cov}^i [\gamma_k^i - \hat{z}_k^-] [\gamma_k^i - \hat{z}_k^-]^T
\]

(3.40)  

(3.41)

The state observation cross covariance matrix
\[
\mathbf{P}_{x_{k-1}}^+ = \sum_{i=0}^{2L} w_{i}^{\text{cov}} [X^i_k - \hat{x}_k^-][\gamma^i_k - \hat{z}_k^-]^T \tag{3.42}
\]

is used to compute the UKF Kalman gain as

\[
K_k = (\mathbf{P}_{x_{k-1}}^+)^{-1} \tag{3.43}
\]

In UKF the updated state is the predicted state plus the innovation weighted by Kalman Gain:

\[
\hat{x}_k^+ = \hat{x}_k^- + K_k \tilde{z}_k^- \tag{3.44}
\]

where \(\tilde{z}_k = z_k - \hat{z}_k^-\) is the residual and the updated covariance is the difference between predicted covariance and the predicted observation covariance as

\[
\mathbf{P}_k^+ = \mathbf{P}_k^- + K_k \mathbf{P}_{\tilde{z}_k^-} K_k^T \tag{3.45}
\]

### 3.4 Constrained Local Sub-Map Filter

Constrained Local Sub-Map Filter (CLSF) [44] is a technique to fuse a local map to a global map. A local map can be estimated using FastSLAM into the whole picture of features in the environment previously estimated by UKF. This technique updates the full covariance matrix of the system generated by UKF to be scheduled at appropriate intervals defined by FastSLAM. This method provides an independent, local sub-map estimate of the point
features in the environment in a small scale and compares the local map that is statistically sampled by particles with the global map produced by UKF-SLAM algorithm. Previous use of such method resulted in a map with the reduction of uncertainty in EKF-SLAM. Instead of using EKF to estimate global and local maps separately and reducing uncertainty, this approach may be substituted by a combination of RBPF and UKF. The observations are fused to create a local map by FastSLAM referenced to a local frame of reference where its global position is already estimated by UKF. At appropriate intervals, the information contained in the local map is transferred into the global map using formulated constraints between the point landmark estimates. The constraint would produce a map of the environment and an estimated path that are identical to the ones previously estimated by UKF. Figure 3.2 illustrates a block diagram of the sub-mapping strategy using RBPF, UKF, and CLSF.

![Figure 3.2: Scheduling of the application on constraints using the CLSF](image-url)
When the vehicle is at location $x^R_k$ at time step $k$, a new frame of reference is initiated. At this moment, the path up to and including time step $k$ is already estimated in the same frame of reference with respect to the global frame of reference and with its minimum uncertainty (assuming it is zero). At the same time step, the global local frame is initialized under UKF calculation. However, the estimation in the local frame of reference by RBPF is completely independent of the estimation that is already done by UKF in the Global frame of reference. At this time step, only a small state covariance matrix of the system in the global frame of reference is updated with the new observation. Prior to the beginning of time step $k+1$, the key is switched off to produce a fused map with minimum uncertainty in the system. Once again, at the beginning of time step $k+1$, the switch turns on to send a local map and to fuse it into the global map estimated by UKF to initiate the process of generating a new global map. The data association block is a simple algorithm to reduce ambiguity of duplicated observations [56].

### 3.4.1 Uncertainty in Position and Observation of Landmarks

With a noisy measurement in motion, the observation distribution for each landmark becomes more uncertain. On the other hand, if the observation is noisy, the observation distribution of nearby landmarks will start to overlap substantially. This overlap causes the ambiguity in recognition of landmarks [57]. Figure 3.3 indicates measurement ambiguity in UHS. In this figure, ellipses illustrate the range of possible observations from two nearby landmarks. In this figure, the dark circle depicts possible observation from either landmark [8].
Figure 4.3: Measurement Ambiguity [8]

Figure 4.4: Motion ambiguity-Uncertainty [8].
While attributing an observation to a wrong landmark will increase error in the map and the pose, its overall impact will be relatively minor. Since the observation could have been generated by either landmark with high probability, the effect of observation of the landmark positions and the robot pose will be minor. The covariance of first landmark will be slightly overestimated while the covariance of the second landmark is slightly underestimated. If multiple observations are incorporated per control, a data association mistake will have relatively little impact on the data association decisions for the other observations [58]. Ambiguity in data association caused by motion noise can have much more severe consequences on the estimation of accuracy. High emotion noise will lead to higher pose uncertainty after incorporating a control. If this pose uncertainty is high enough, having different robot poses in this distribution will drastically imply different ML data association hypothesis for subsequent observations. In figure 3.4, it is shown that the motion ambiguity is easily induced if there is significant rotational error in the motion of robot. Moreover, if multiple observations are incorporated per control, robot’s pose will correlate the data association decisions of all of the observations. If the SLAM algorithm chooses the wrong data associations for a single observation, the rest of the data associations will also be incorrect [59].

### 3.4.2 Reducing Uncertainty

Figure 3.5 demonstrates a robot that is traveling through an environment observing two landmarks where at time step $k-1$, the sensor is observing landmarks $i$ and $j$. UKF-SLAM sufficiently carries a huge uncertainty over both observed landmarks. As the robot travels from time step $k-1$ to time step $k$, landmarks are observed and mapped using RBPF. Due to
the property of FastSLAM, the uncertainty over the probability distribution of landmarks locations is much less than uncertainty of the map produced by UKF. Next step is to estimate the location of landmarks using RBPF, and then, through the reducing error mechanism of CLSF, the uncertainty of the map reduces. Once the global map is replaced by the local map at time step $k$, as the robot observes more landmarks, the uncertainty of the map is reduced and the result is a consistent and converged map [60].

![Figure 3.5: Reduction of uncertainty over location of a landmark](image)

### 3.4.3 Adding Landmarks to the Local Map

Since more landmarks are observed in next time steps, there is a need for a proper formulation in order to make the mapping process reliable [61]. If a specific measurement is insufficient to constrain the new landmark in all dimensions, the decision to add a new landmark could be
quite difficult. In Kalman-based algorithms, the measurement function $h(x_k^R, z_k)$ is invertible, meaning that only a single measurement is made to initialize a new landmark. In the RBPF, each observation may be represented in a Gaussian form as [54]

$$z_k \sim N(\hat{z}_k + \Lambda_k (z_{k,i} - n \mu_{k-1,i}), R_k) \quad (3.46)$$

This Gaussian can be written as

$$\frac{1}{\sqrt{2\pi |Z_{k,i}|}} \exp\left[-\frac{1}{2}(z_k - \hat{z}_k - \Lambda_k (z_{k,i} - n \mu_{k-1,i}))^T R_k^{-1} (z_k - \hat{z}_k - \Lambda_k (z_{k,i} - n \mu_{k-1,i}))\right] \quad (3.47)$$

Q is defined as a function equal to the negative of the exponent of this Gaussian:

$$Q = \frac{1}{2} (z_k - \hat{z}_k - \Lambda_k (z_{k,i} - n \mu_{k-1,i}))^T R_k^{-1} (z_k - \hat{z}_k - \Lambda_k (z_{k,i} - n \mu_{k-1,i})) \quad (3.48)$$

The second derivative of Q with respect to $z_{k,i}$ will be the inverse of the covariance matrix of the Gaussian in landmark coordinates.

$$\frac{\partial Q}{\partial z_{k,i}} = - (z_k - \hat{z}_k - \Lambda_k (z_{k,i} - n \mu_{k-1,i}))^T R_k^{-1} \Lambda_k \quad (3.49)$$

$$\frac{\partial^2 Q}{\partial z_{k,i}^2} = \Lambda_k^T R_k^{-1} \Lambda_k \quad (3.50)$$
As a result, an invertible observation can be used to create a new landmark as

\[ n \mu_{k-1,i} = h^{-1}(n_\mu x_k^R, z_k) \]  
(3.51)

\[ n \Sigma_{k,i} = \Lambda_k^T R_k^{-1} \Lambda_k \]  
(3.52)

\[ n \hat{\omega}_k = P_0 \]  
(3.53)

The initialization of landmarks may be calculated through a simpler formulation. By setting the variance of each landmark parameter to a high value and incorporating the initial observation, the exact initial covariance does not have to be considered. Higher KG values lead the process to a more accurate approximation regarding the observation of each landmark [30].

\[ n \mu_{k-1,i} = h^{-1}(n_\mu x_k^R, z_k) \]  
(3.54)

\[ n \Sigma_{k,i} = K I \]  
(3.55)

### 3.4.4 State and Covariance Matrices after Fusion

The Unscented HybridSLAM algorithm is arranged in such way so that UKF-SLAM estimates the whole map of the environment and the RBPF estimates the path and the local map in the vicinity of the current robot position. At the CLSF part, there will be an update only on features that are observed in the current local frame of reference, and the remaining map information will be untouched. When the information in the local map is fused into the global map, the resulting map is replaced with the map in the previous time step. There are
two distinct state estimates independent from each other, and as a result, the augmented form of posterior state in the process of map fusion can be expressed as

\[
\hat{x}_k^{+}\text{(CLSF)} = \{ \hat{x}_k^{+}\text{(robot)}, \hat{m}_k^{+}, \hat{x}_k^{+}\text{(robot)}, \hat{m}_k^{+} \}
\] (3.56)

where, \(\hat{x}_k^{+}\text{(robot)}\) is the global posterior position of the robot estimated by UKF, \(\hat{x}_k^{+}\text{(robot)}\) is the local posterior position of the robot estimated by RBPF, \(\hat{m}_k^{+}\) is the map of landmarks estimated by UKF and \(\hat{m}_k^{+}\) is the map of landmarks in the vicinity of the robot’s pose only and estimated by RBPF. The system covariance is defined as

\[
P_k^{+} = \begin{bmatrix}
G \hat{P}_k^{+} & 0 \\
0 & L \hat{P}_k^{+}
\end{bmatrix}
\] (3.57)

\[
G \hat{P}_k^{+} = \begin{bmatrix}
G \hat{P}_k^{+} & G \hat{P}_k^{+} \\
G \hat{P}_k^{+} & mL \hat{P}_k^{+}
\end{bmatrix}
\] (3.58)

\[
L \hat{P}_k^{+} = \begin{bmatrix}
L \hat{P}_k^{+} & L \hat{P}_k^{+} \\
L \hat{P}_k^{+} & mR \hat{P}_k^{+}
\end{bmatrix}
\] (3.59)

\(L \hat{P}_k^{+}\) is the robot covariance in the local frame of reference, \(mL \hat{P}_k^{+}\) indicates covariance in the local frame of reference related to landmarks, \(mR \hat{P}_k^{+}\) and \(mL \hat{P}_k^{+}\) represent covariance on robots and landmarks in the local frame of reference as well. \(G \hat{P}_k^{+}\), \(G \hat{P}_k^{+}\), and \(mL \hat{P}_k^{+}\) represent same
covariance matrices as above but in the global frame of reference. Finally, $\mathbf{P}_k^G$ is the covariance of the estimate of the local frame of reference with respect to the global frame of reference. In this approach, the position of the robot and the map in the vicinity of the robot position is estimated using particle filters. The resulted data is converted to a single Gaussian and by the use of CLSF the estimated local map is fused to the Global map previously estimated by UKF. The position of the robot estimated by RBPF in the local map is considered to be as an additional landmark [62]. When the local map covariance matrix is fused to the global map, the related data stays in the main covariance matrix until the next time step.

3.5 Solution to Sample Impoverishment

The main reason for the failure of a particle-based filter is when the system faces a noisy motion with an accurate observation sensor. In this case, the proposal distribution does not match the target distribution, and most of particles will be dismissed in the re-sampling process and not incorporated in the estimation of neither robot path nor the environment map. In other words, if there is not a balance between distributions in the re-sampling step, the amount of diversity increases. In such cases there will be a need for many more particles so that, subsequent to the re-sampling step, a considerable amount of particles remain in the estimation process. However, increasing number of particles has some consequences where at the outset, there will be a need for a huge amount of memory which increases calculation costs. Furthermore, if there is a loop closing scenario or a very large environment with too many landmarks, there will be a possibility that at the end of each loop the remaining particles are withdrawn. To match both distributions, a new representation of particles in the sub-filter
is proposed in [8]. In this modification, drawing sample procedure is developed such that particle drawing is monitored and in the update process, the weight of incorporating particles does not gain a low value [63].

### 3.5.1 Proposal Distribution

Since the entire Unscented HybridSLAM presentation is in a Gaussian form, normalizing non-linear functions of motion and observation models will properly match the motion and observation models into linear functions [8]. Therefore, it is fair to represent motion and observation models as

\[
P(\mathbf{x}_k^R, \mathbf{m} | \mathbf{u}_k, \mathbf{x}_{k-1}^R) = f(\mathbf{x}_{k-1}^R, \mathbf{u}_k) + \mathbf{\hat{w}}_k
\]

\[
\mathbf{\hat{w}}_k \sim N(0, \mathbf{\hat{Q}}_k)
\]

\[
\mathbf{\hat{Q}}_k = E[(\mathbf{\hat{w}}_k)(\mathbf{\hat{w}}_k)^T]
\]

\[
P(\mathbf{z}_k | \mathbf{x}_k^R, \mathbf{m}, \mathbf{d}_k) = h(\mathbf{x}_k^R, \mathbf{z}_{k,i}) + \mathbf{\hat{v}}_k
\]

\[
\mathbf{\hat{v}}_k \sim N(0, \mathbf{R}_k)
\]

\[
\mathbf{R}_k = E[(\mathbf{\hat{v}}_k)(\mathbf{\hat{v}}_k)^T]
\]

The sub-filter formulation is therefore, changed based on probability of the new system pose at the current time step conditioned on the estimated path up to, and not including the current time step, the most recent observations, the control action that leads the robot to the new pose, and finally, the data association. The new pose is expressed as
\[ n x^R_k \sim P(x_k | x^R_{k-1}, U_k, Z_k, d_k) \]  \hspace{1cm} (3.66)

To sample \( n x^R_k \), motion model, observation model, and the Gaussian feature estimates must be considered. Using Bayes rule, the proposal distribution can be expanded as

\[ n x^R_k \sim P(x_k | x^R_{k-1}, U_k, Z_k, d_k) \]  \hspace{1cm} (3.67)

\[
\begin{align*}
&\text{Bayes} \\
&\propto \eta P(z_k | x^R_k, x^R_{k-1}, U_k, Z_{k-1}, d_k) P(x_k | x^R_{k-1}, U_k, Z_k, d_k) \\
&= \eta P(z_k | x^R_k, x^R_{k-1}, U_k, Z_{k-1}, d_k) P(x_k | x^R_{k-1}, U_k, Z_k, d_k)
\end{align*}
\]  \hspace{1cm} (3.68)

This equation can be further simplified through Markov assumption which states that given knowledge of the previous path \( x^R_{k-1} \), it is of no importance how the robot estimated that path and what observations were involved up to that time. Therefore equation (3.67) becomes

\[ n x^R_k \sim P(x_k | x^R_{k-1}, U_k, Z_k, d_k) \]  \hspace{1cm} (3.69)

Furthermore, with the use of total probability theorem, equation (3.69) can be more simplified when observing landmark \( m_j \) and expressed as

\[ n x^R_k \sim P(x_k | x^R_{k-1}, U_k, Z_k, d_k) \]  \hspace{1cm} (3.70)
Applying Markov assumption for equation (3.70) leads to the final representation of the posterior pose in equation (3.71) in which the sampling distribution is the convolution of two Gaussians, multiplied by another Gaussian. In this equation the first term represents the observation model same as [64].

\[ n x_k^R \sim P(x_k | x_{k-1}^R, U_k, Z_k, d_k)^{\text{Markov}} = \eta P(z_k | m_j, x_k^R, d_k) P(m_i | x_{k-1}^R, U_k, Z_{k-1}, d_k) \, dm_i \, P(x_k | x_{k-1}^R, u_k) \] (3.71)

Each term in above equation can be expressed with a normal distribution as follows

\[ P(z_k | m_i, x_k^R, d_k) \sim N(z_k, h(x_k^R, z_{k,i}), R_k) \] (3.72)

\[ R_k = E[(\hat{v}_{k,i})(\hat{v}_{k,i})^T] \] (3.73)

\[ P(m_i | x_{k-1}^R, U_k, Z_{k-1}, d_k) \sim N(m_j, h^{-1}(x_{k-1}^R, z_{k,j}), \Sigma_{k-1,j}) \] (3.74)

\[ \Sigma_{k,j} = \Lambda_k^T R_k^{-1} \Lambda_k \] (3.75)

\[ P(x_k | x_{k-1}^R, u_k) \sim N(x_k, f(x_{k-1}^R, u_k), \hat{Q}_k) \] (3.76)

\[ \hat{Q}_k = E[(\hat{w}_k)(\hat{w}_k)^T] \] (3.77)
Since function $h$ is not linear, the distribution has no closed form and samples cannot be drawn simply. Therefore, the observation function has to be presented in a linear form so that samples can be drawn. For the linearization, the first order of Taylor series is considered and the observation function is represented as

\[
\dot{x}_k^R = f (^n x_{k-1}^R, u_k) 
\]

(3.78)

\[
\dot{z}_k = h (\dot{x}_k^R, n \mu_{k-1,i})
\]

(3.79)

\[
\Lambda_k = \frac{\partial h(x)}{\partial z_{k,i}} \bigg|_{x=x_k^R, z_{k,i} \in \mu_{k-1,i}} \quad \text{where } x_k^R \text{ is replaced by } \dot{x}_k^R
\]

(3.80)

\[
\Lambda_k^{\text{position}} = \frac{\partial h(x)}{\partial x_k^R} \bigg|_{x=x_k^R, z_{k,i} \in \mu_{k-1,i}} \quad \text{where } x_k^R \text{ is replaced by } \dot{x}_k^R
\]

(3.81)

\[
h(x_k^R, z_{k,i}) \approx h (\dot{x}_k^R, n \mu_{k-1,i}) + \Lambda_k [mj - h^{-1} (n x_{k-1}^R, z_{k,j})] + \Lambda_k^{\text{position}} (x_k^R - \dot{x}_k^R)
\]

(3.82)

where $\dot{x}_k^R$ is the predicted pose at time step $k$ and $\dot{z}_k$ is the residual or predicted observation after incorporating all observations up to and including time step $k$. $\Lambda_k$ and $\Lambda_k^{\text{position}}$ are Jacobians of function $h(.)$ and are derivatives of the function with respect to observation and pose respectively. An approximation of the proposal distribution can now be modeled as a normal distribution function as

\[
N (\dot{z}_k, \dot{z}_k + \Lambda_k^{\text{position}} \dot{x}_k^R, \dot{R}_k)
\]

(3.83)

\[
\dot{R}_k = R_k + \Lambda_k n \Sigma_{k-1,i} \Lambda_k^T
\]

(3.84)
Expanding the form of Gaussian explicitly, equation (4.53) and its first and second derivatives can be written as

\[ P(x_k | \begin{array}{c} X_k^{R-1}, U_k, Z_k, d_k \end{array}) = \xi \exp \{ -y_k \} \]  
\[ y_k = \frac{1}{2} \left[ (z_k - \hat{z}_k - \Lambda_k^{\text{position}} x_k^R + \Lambda_k^{\text{position}} \hat{x}_k^R)^T \hat{R}_k^{-1} (z_k - \hat{z}_k - \Lambda_k^{\text{position}} x_k^R + \Lambda_k^{\text{position}} \hat{x}_k^R) + \right. \]
\[ \left. (x_k^R - \hat{x}_k^R)^T \hat{Q}_k^{-1} (x_k^R - \hat{x}_k^R) \right] \]  
\[ \frac{\partial y_k}{\partial x_k^R} = - (\Lambda_k^{\text{position}})^T \hat{R}_k^{-1} (z_k - \hat{z}_k - \Lambda_k^{\text{position}} x_k^R + \Lambda_k^{\text{position}} \hat{x}_k^R) + \hat{Q}_k^{-1} (x_k^R - \hat{x}_k^R) \]  
\[ = \left[ (\Lambda_k^{\text{position}})^T \hat{R}_k^{-1} \Lambda_k^{\text{position}} \right] x_k^R - (\Lambda_k^{\text{position}})^T \hat{R}_k^{-1} (z_k - \hat{z}_k + \Lambda_k^{\text{position}} \hat{x}_k^R) \]  
\[ \frac{\partial^2 y_k}{(\partial x_k^R)^2} = \left[ (\Lambda_k^{\text{position}})^T \hat{R}_k^{-1} \Lambda_k^{\text{position}} \right] \]  

The second derivative in equation (3.89) is introducing the covariance and mean of sampling distribution as

 Sampling Distribution Covariance = \left[ (\Lambda_k^{\text{position}})^T \hat{R}_k^{-1} \Lambda_k^{\text{position}} \right]^{-1} \]  
\[ \text{Sampling Distribution Mean} = \]
\[ \left[ (\Lambda_k^{\text{position}})^T \hat{R}_k^{-1} \Lambda_k^{\text{position}} \right]^{-1} \left[ (\Lambda_k^{\text{position}})^T \hat{R}_k^{-1} (z_k - \hat{z}_k) \right] + \hat{x}_k^R \]
3.5.2 Updating Importance Weight

Based on the new proposal distribution formulation, the importance weight step needs to be reformulated according to [8]. The new formulation is defined as the target distribution ratio over the proposal distribution, and under the correction assumption that the path $^n x^R_{k-1}$ was estimated for a particle “n” according to the target distribution at the previous time step. The proposal distribution is already given by

$$P(^n X^R_{k-1} | U_{k-1}, Z_{k-1}, d_{k-1}) P(^n x^R_k | ^n X^R_{k-1}, U_k, Z_k, d_k)$$

(3.92)

Since the importance weight is equal to the target distribution over the proposal distribution it can be calculated as

\[
^n \hat{W}_k = \frac{\text{Target distribution}}{\text{Proposal distribution}} = \frac{P(^n X_k | Z_k, U_k, d_k)}{P(^n X_{k-1} | Z_{k-1}, U_{k-1}, d_{k-1}) P(^n x^R_k | ^n X^R_{k-1}, Z_k, U_k, d_k)}
\]

(3.93)

Using the concept of conditional probability, the importance weight described in equation (3.93) can expand to

\[
^n \hat{W}_k = \frac{P(^n X_k | ^n X^R_{k-1}, Z_k, U_k, d_k) P(^n X_{k-1} | Z_{k-1}, U_{k-1}, d_{k-1})}{P(^n X_{k-1} | Z_{k-1}, U_{k-1}, d_{k-1}) P(^n x^R_k | ^n X^R_{k-1}, Z_k, U_k, d_k)}
\]

(3.94)

With the use of Bayes rule and Markov assumption
\[
\hat{W}_k^\text{Bayes} = \eta \frac{P(z_k^n | x^R_{k-1}, z_{k-1}, U_k, d_k)P(x^R_{k-1} | Z_{k-1}, U_k, d_k)}{P(x^R_{k-1} | Z_{k-1}, U_k, d_k)}
\] (3.95)

\[
\hat{W}_k^\text{Markov} = \eta \frac{P(z_k^n | x^R_{k-1}, z_{k-1}, U_k, d_k)P(x^R_{k-1} | Z_{k-1}, U_k, d_k)}{P(x^R_{k-1} | Z_{k-1}, U_k, d_k)} = \eta P(z_k^n | x^R_{k-1}, Z_{k-1}, U_k, d_k)
\] (3.96)

Now the theorem of Total Probability can be applied twice to express the importance weight as a conditional probability on the augmented system vector as below

\[
\hat{W}_k = \eta \int P(z_k^n | x^R_{k-1}, z_{k-1}, U_k, d_k)P(x^R_{k-1} | Z_{k-1}, U_k, d_k)dx^R_k
\] (3.97)

\[
\hat{W}_k^\text{Markov} = \eta \int P(z_k^n | x^R_{k-1}, z_{k-1}, U_k, d_k)P(x^R_{k-1} | Z_{k-1}, U_k)dx^R_k
\] (3.98)

\[
= \eta \iint P(z_k^n | x_k, z_{k-1}, U_k, d_k)P(m_i | x^R_{k-1}, z_{k-1}, U_k)dm_iP(x^R_{k-1} | Z_{k-1}, U_k)dx^R_k
\] (3.99)

Applying Markov assumption one more time for equation (3.97) results in

\[
\hat{W}_k = \eta \iint P(z_k^n | x_k, d_k)P(m_i | x^R_{k-1}, U_k)dm_iP(x^R_{k-1} | Z_{k-1}, U_k)dx^R_k
\] (3.100)

All three terms in equation (3.100) are Gaussian distributed. The first term corresponds to the observation model, the second term corresponds to the landmark estimator, and the third term simply describes a probability distribution function of the motion model.
\[ P(z_k | x_k, d_k) \sim N(\mathbf{z}_k, h(x_k^R, \mathbf{z}_{k,i}), \mathbf{R}_k) \] (3.101)

\[ \mathbf{R}_k = E[(\hat{\mathbf{y}}_{k,i})(\hat{\mathbf{y}}_{k,i})^T] \] (3.102)

\[ P(m_i | n \mathbf{X}_{k-1}, \mathbf{Z}_{k-1}, \mathbf{U}_{k-1}, d_{k,i}) \sim N(\mathbf{m}_i, h^{-1}(n \mathbf{x}_{k-1}^R, \mathbf{z}_{k,i}), n \Sigma_{k-1,i}) \] (3.103)

\[ n \Sigma_{k,i} = \Lambda_k^T \mathbf{R}_k^{-1} \Lambda_k \] (3.104)

\[ P(x_k^R | n \mathbf{X}_{k-1}, \mathbf{U}_k) \sim N(x_k^R, f(n \mathbf{x}_{k-1}^R, u_k), \hat{Q}_k) \] (3.105)

\[ \hat{Q}_k = E[(\hat{\mathbf{w}}_k)(\hat{\mathbf{w}}_k)^T] \] (3.106)

As a result, the importance weight may be expressed in a Gaussian form with a mean and covariance as

\[ n \hat{\mathbf{w}}_k \sim N(\mathbf{z}_k, \tilde{\mathbf{z}}_k, \tilde{\mathbf{Z}}_{k,i}) \] (3.107)

\[ \tilde{\mathbf{Z}} = (\Lambda_k^{\text{position}})^T \mathbf{R}_k^{-1} \Lambda_k^{\text{position}} + \Lambda_k^T n \Sigma_{k-1,i} \Lambda_k + \mathbf{R}_k \] (3.108)

and

\[ n \hat{\mathbf{w}}_k = \frac{1}{\sqrt{|2\pi\tilde{\mathbf{Z}}_{k,i}|}} \exp\left(-\frac{1}{2} \tilde{\mathbf{z}}_{k,i}^T \tilde{\mathbf{Z}}_{k,i}^{-1} \tilde{\mathbf{z}}_{k,i}\right) \] (3.109)

Based on the new formulation of the proposal distribution and importance sampling, the algorithm can be reformulated [8]. The algorithm is illustrated in appendix C.
3.6 Data Association

UKF algorithm based on particle filtering pose estimation, takes a multi hypothesis approach to the data association problem where each particle represents a different hypothesized path. Therefore, data association decisions can be made on a per-particle basis. Particles that earn the correct data association will receive high weights since they explain the observations well, and particles that earn wrong associations will receive low weights and are removed in a future re-sampling step [8].

3.6.1 Data Association for Each Particle

Per particle-data association has several important advantages over standard MLE data association. First, it factors the pose uncertainty out of the data association problem. Since the motion ambiguity is the more severe form of data association ambiguity, conditioning the data association decisions on hypothesized paths appears to be a logical decision. In such a case, some of the particles would draw new positions consistent with the data association on the top, while others would draw positions consistent with the data association hypothesis on the bottom. Additionally, unlike Kalman-based algorithms, UKF represents the data association and robot pose uncertainties by the entire particle set. The landmark filters in a single particle are not affected by motion noise due to the fact that they are conditioned on a specific robot path. This is especially useful if the robot has noisy motion and an accurate sensor [29].

The simplest approach to per-particle data association is to apply MLE data association heuristic, only on a per-particle basis. Since the landmark estimators are based on EKF, the likelihood can be calculated using innovations. If the value of this likelihood falls below some threshold $P_0$, a new landmark is added to the particle [30].
Maximum Likelihood Estimator

Maximum Likelihood Estimator (MLE) data association tends to work much more reasonable in RBPF than it does in KF-based approaches due to the fact that the most severe component of data association ambiguity originates from uncertainty in the pose. Some fraction of the particles will draw new poses consistent with the true robot pose. These poses will receive correct data associations and explain the observations well and particles that draw poses far from the true pose will receive wrong data associations that explain the data poorly.

In practice, the number of landmarks in the map cannot be obtained trivially [8]. Instead, the data association has to be solved between the observation of landmarks at time step \( k \) and set of landmarks in the map. To solve this problem, Maximum Likelihood Estimator is employed. If \( d_{k,i} \) is the data association where \( i \in \{1, 2, \ldots, M\} \), the probability of \( d_{k,i} \) can be described [8] by

\[
P(\mathbf{d}_{k,i} | \mathbf{Z}_k, \mathbf{U}_k) = \int P(\mathbf{d}_{k,i} | \mathbf{X}_k, \mathbf{Z}_k, \mathbf{U}_k) P(\mathbf{X}_k | \mathbf{Z}_k, \mathbf{U}_k) \, d\mathbf{X}_k
\]

(3.111)

\[
\text{RBPF} \approx \sum_n P(\mathbf{d}_{k,i} | n \mathbf{X}_k, \mathbf{Z}_k, \mathbf{U}_k)
\]

(3.112)

\[
\text{Markov} = \sum_n P(\mathbf{d}_{k,i} | n \mathbf{X}_k, \mathbf{Z}_k)
\]

(3.113)

\[
\text{Bayes} \propto \sum_n P(\mathbf{z}_k | n \mathbf{x}_k, \mathbf{d}_{k,i})
\]

(3.114)
When $d_{k,i}$ in equation (3.114) approaches its maximum value, it represents the maximum likelihood data association. When the maximum value of data association probability in equation (3.111) is below a threshold $\alpha$, the landmark is considered previously unseen and the map is augmented accordingly [32]. Due to its RBPF sub-filter property, UKF does not determine the data association once per each observation and therefore it carries a multiple-hypothesis data association. As a result, if a wrong landmark observation is inaccurate, the particle does not gain enough weight in the re-sampling process and the related data is blocked. Only particles with enough weight in re-sampling step pass the filter and are incorporated to the local map.

### 3.6.3 Elimination of Wrong Observations

The environment containing landmarks is always considered as an empty 2D space unless a landmark is observed and registered. Assume that at time step $k$, the algorithm has not included any feature in the local map. Once a landmark is detected and observed, a positive evidence of existence of that specific landmark is used as data to be considered in the environment. In the next time step $k+1$, the landmark is re-observed and the task of the algorithm is now to determine if the observed landmark is exactly the one observed in the previous time step $k$. If there is not such mechanism included in the algorithm, a wrong landmark may be added to the map due to a particular wrong observation. The result will then be a point occupied in the environment map and the consistency of the filter will be under question [65].

To include such a monitoring mechanism in the algorithm, a “negative evidence sub-algorithm” same as [36] is used for the UHS. Negative evidence sub-algorithm determines
whether an observed landmark actually exists in the environment or not. For the sub-algorithm, one considers a probability function of the binary variable $i_L^k$ to indicate whether the landmark $m_i$ is real or false. The function measures the probability of landmark existence conditioned on path, observation, and the data association corresponding to the current time step described as

$$P(i_L^k | Z_k, U_k, d_{k,i})$$

(3.115)

The representation of this probability function is compatible with a Bayesian filter described in chapter 1. With a log-odds representation, the posterior over existence of the landmark can be written as

$$\mu = \ln \frac{P(i_L^k | Z_k, X_k, d_{k,i})}{1 - P(i_L^k | Z_k, X_k, d_{k,i})} = \sum_k \ln \frac{P(i_L^k | z_k, x_k, d_{k,i})}{1 - P(i_L^k | z_k, x_k, d_{k,i})}$$

(3.116)

Since the above representation is in form of log-odds, they have the advantage of being added to the algorithm [66]. If a landmark observation gains a positive value, it will be added to, and if negative it will be deducted from $\mu$. The positive and negative values can be implemented in real time similar to [8] and are defined as

$$\text{Value}^+ = \ln \frac{P(i_L^k | z_k, x_k, d_{k,i})}{1 - P(i_L^k | z_k, x_k, d_{k,i})}$$

(3.117)

$$\text{Value}^- = \ln \frac{P(i_L^k | z_k, x_k, d_{k,i})}{1 - P(i_L^k | z_k, x_k, d_{k,i})}$$

(3.118)
If the landmark is observed, positive or negative values are added to or subtracted from $\mu$ depending on the probability of the value. If $\mu$ falls below a minimum value corresponding to the minimum probability of being at that specific area of observation, then the landmark is considered nonexistent. Otherwise, the values are added or deducted if observed or missed [8].

3.7 Logarithmic Computational Complexity

In the logarithmic computational complexity [8], the linearity of $M$ landmark filters that are duplicated at each time step will be shared among particles. As a result, the FastSLAM as part of UKF algorithm follows the same precision as particle filter in building its local map. To convert the linearity of $M$ filters to a logarithmic form, the particle representation may be expressed as an array of landmark filters to a binary tree as indicated in figure 4.6. In this figure, ‘$i$’ identifies a randomly chosen landmark, and a binary tree for eight landmark filters is shown. Any sub-tree can be shared among many landmarks. While this representation is relatively complicated, a lot of memory for the computation will be saved [35]. On average, the result of such tree will end up with a logarithmic complexity in computation. As a result, the computation complexity will be calculated as $P \times \log(M)$ which become less complex compared to the regular HybridSLAM algorithm.
Figure 3.6: The binary tree of landmark filters [8]

Figure 3.7: The landmark tree update procedure [8]
Figure 4.7 depicts a specific situation in which the third landmark at time step \( k \) is observed and updated. In this incomplete tree, only the third landmark Gaussian parameters are updated and instead of the entire tree, a single path from the root to the third landmark Gaussian parameters is duplicated. The tree is completed by copying the missing pointers from the generating particles tree, thus, branches that leave the modified path will point to the unmodified sub-trees of the generating particle [8].

It appears that generating the modified tree takes time logarithmic in \( M \). Moreover accessing a Gaussian also takes time logarithmic in \( M \). The number of steps needed for navigating to a leaf of the tree is equivalent to the length of the path. Therefore, both generating and accessing a partial tree can be done in a logarithmic form. Since at every update step \( M \) new particles are generated, the FastSLAM algorithm requires time \( P \times \log(M) \).

### 3.8 Proof of Filter Convergence [8]

Based on Montemerlo [8], with the logarithmic presentation of the filter and based on reformulation of the proposal distribution and importance weight, the landmark position estimating update equations can be rearranged as

\[
\hat{x}_k = f(n^{R}_x \hat{x}_{k-1}, u_k) = n^{R}_x \hat{x}_{k-1} + u_k
\]  

(3.119)

\[
\hat{z}_k = h(\hat{x}_k, h^{-1}(n^{R}_x \hat{x}_{k-1}, \hat{z}_{k,j})) = h^{-1}(n^{R}_x \hat{x}_{k-1}, \hat{z}_{k,j}) - n^{R}_x \hat{x}_{k-1} - u_k
\]  

(3.120)

\[
\Lambda_k = \frac{\partial h(x)}{\partial z_{k,i}} \bigg|_{x = n^{R}_x \hat{x}_{k-1}, \hat{z}_{k,j} = n^{R}_x \hat{x}_{k-1}, u_k} \text{ where } n^{R}_x \hat{x}_{k-1} \text{ is replaced by } \hat{x}_k
\]  

(3.121)

\[
\Lambda_k^{position} = \frac{\partial h(x)}{\partial x_k} \bigg|_{x = n^{R}_x \hat{x}_{k-1}, \hat{z}_{k,j} = n^{R}_x \hat{x}_{k-1}, u_k} \text{ where } n^{R}_x \hat{x}_{k-1} \text{ is replaced by } \hat{x}_k
\]  

(3.122)
\[ \hat{R}_k = R_k + \Lambda_k \sum_{k-1,i} \Lambda_k^\top = \hat{R}_k = R_k + \sum_{k-1,i} \] (3.123)

\[ \Sigma_k^{\text{Position}} = \left[ \left( \Lambda_k^{\text{position}} \right)^\top \hat{R}_k \Lambda_k^{\text{position}} \right] + \hat{Q}_k^{-1} \] (3.124)

\[ \mu_k^{\text{Position}} = \Sigma_k^{\text{Position}} \Lambda_k^{\text{position}} \hat{R}_k^{-1} (z_k - \hat{z}_k) + \hat{x}_k \] (3.125)

\[ = -\Sigma_k^{\text{Position}} (R_k + \sum_{k-1,i} \sum_{k-1,i}^{-1} (z_k - h^{-1}(n_{x_{k-1,i},z_{k,j}} + n_{x_{k-1,i}} + u_k) + n_{x_{k-1,i}} + u_k) \] (3.126)

\[ n_{x_k}^{R} \sim N(x_k, \mu_k^{\text{Position}}, \Sigma_k^{\text{Position}}) \] (3.127)

\[ \mu_k^{\text{Landmarks}} = h^{-1}(x_{k-1,i},z_k) + \sum_{k-1,i} (R_k + \sum_{k-1,i}^{-1} (z_k - h^{-1}(n_{x_{k-1,i},z_{k,j}} + n_{x_{k-1,i}} + u_k) \] (3.128)

\[ \Sigma_k^{\text{Position}} = (A_k - \sum_{k-1,i} (R_k + \sum_{k-1,i}^{-1}) \sum_{k-1,i} \] (3.129)

As discussed in [8], a series of lemmas are carried through the convergence proof in which variable \(^n\alpha_k\) is assumed to be the absolute error on the position of the robot and \(^n\beta_k\) is considered as a variable for absolute error in the location of landmark \(i\) at time step \(k\). The first lemma characterizes the effect of \(^n\beta_k\) on \(^n\alpha_k\) where

\[ ^n\alpha_k = n_{x_k}^{R} - x_k^{R} \] (3.130)

\[ ^n\beta_k = ^n\mu_{k,i}^{\text{Landmarks}} - m_{k,i} \] (3.131)

In next subsections of this section the same approach in [8] is shown to prove the FastSLAM filter convergence. The filter convergence becomes visibly evident in chapter 6 with the appearance of successive simulations.
3.8.1 Lemma 1

If the observation vector of landmark $m_{k,i}$ at time step $k$ is $z_{k,i}$, and the error in observation of that landmark is $\beta_{k,i} \leq \alpha_k$ in terms of its magnitude, $\alpha_k$ decreases in expectation as a result of the observation [35]. If $\beta_{k,i} > \alpha_k$, the error in the position of the robot may increase but the expectation will not exceed the error in estimating the landmark position [8].

**Proof** – The expected value on the pose error is

$$E[\alpha_k] = E[\mathbf{x}_k^R - \mathbf{x}_k^R] = E[\mathbf{x}_k^R] - E[\mathbf{x}_k^R]$$  \hspace{1cm} (3.132)

$$= E[-\mathbf{z}_{\mathbf{R}}(\mathbf{R}_k + n \sum_{k-1}^{-1})^{-1}] (z_k - h^{-1}(\mathbf{x}_{k-1}^R, z_{k,i}) + n \mathbf{x}_{k-1}^R + u_k) + n \mathbf{x}_{k-1}^R + u_k]$$

$$- E[\mathbf{x}_{k-1}^R + u_k]$$  \hspace{1cm} (3.133)

$$= -\mathbf{z}_{\mathbf{R}}(\mathbf{R}_k + n \sum_{k-1}^{-1})^{-1} (E[z_k] - h^{-1}(\mathbf{x}_{k-1}^R, z_{k,i}) + n \mathbf{x}_{k-1}^R + u_k)$$

$$+ (n \mathbf{x}_{k-1}^R - \mathbf{x}_{k-1}^R)$$  \hspace{1cm} (3.134)

where $n \mathbf{x}_{k-1}^R - \mathbf{x}_{k-1}^R$ is the absolute error on the position of the robot $\alpha_{k-1}$ at time step $k-1$, and the final transformation indicates the linearity of the expectation. In linear Gaussian SLAM, the expectation of observation $z_{k,i}$ is expressed as

$$E[z_{k,i}] = m_{k,i} - E[\mathbf{x}_k^R] = m_{k,i} - E[\mathbf{x}_k^R] - \mathbf{x}_{k-1}^R - u_k$$  \hspace{1cm} (3.136)

therefore
\[ E[z_{k,i}] - h^{-1}(^{n}x_{k-1}^{R},z_{k,j}) + \alpha_{k-1}^{n} + u_{k} = m_{k,i} - u_{k} - x_{k-1}^{R} - h^{-1}(^{n}x_{k-1}^{R},z_{k,j}) + \alpha_{k-1}^{n} + u_{k} \]  
(3.137)

\[ = (^{n}x_{k-1}^{R} - x_{k-1}^{R}) + (m_{k,i} - h^{-1}(^{n}x_{k-1}^{R},z_{k,j})) \]  
(3.138)

\[ = \alpha_{k-1}^{n} - \beta_{k-1}^{n} \]  
(3.139)

Recalling equation (3.124) and (3.135), the expected value of the absolute error on the robot position at time step \( k \) may be expressed as

\[ E[\alpha_{k}] = \alpha_{k-1}^{n} + \Sigma_{k}^{Position} \left( R_{k} + \Sigma_{k-1,i}^{n} \right)^{-1} (\beta_{k-1}^{n} - \alpha_{k-1}^{n}) \]  
(3.140)

\[ = \alpha_{k-1}^{n} + \left[ \left( R_{k} + \Sigma_{k-1,i}^{n} \right)^{-1} + \hat{Q}_{k}^{-1} \right]^{-1} \left( R_{k} + \Sigma_{k-1,i}^{n} \right)^{-1} (\beta_{k-1}^{n} - \alpha_{k-1}^{n}) \]  
(3.141)

\[ = \alpha_{k-1}^{n} + \left[ \Lambda_{k} + \left( R_{k} + \Sigma_{k-1,i}^{n} \right) \hat{Q}_{k}^{-1} \right]^{-1} (\beta_{k-1}^{n} - \alpha_{k-1}^{n}) \]  
(3.142)

\( R_{k}, \Sigma_{k-1,i}^{n} \) and \( \hat{Q}_{k}^{-1} \) are all positive semi-definite. Therefore, \( \left[ \Lambda_{k} + \left( R_{k} + \Sigma_{k-1,i}^{n} \right) \hat{Q}_{k}^{-1} \right]^{-1} \) is a positive semi-definite, with its eigenvalues less than 1. Hence, this observation effectively proves lemma 1. In particular, \( \alpha_{k}^{n} \) decreases if \( \beta_{k-1}^{n} < \alpha_{k-1}^{n} \) in terms of magnitude and if \( \alpha_{k-1}^{n} < \beta_{k-1}^{n} \), then \( \alpha_{k}^{n} \) will increase in expectation. However, the increase in expectation value is proportional to the difference ensuring expectation of \( \alpha_{k}^{n} \leq \beta_{k}^{n} \) [8].

### 3.8.2 Lemma 2

If landmarks are all stationary, the absolute error on the robot position will shrink in expectation [8].

**Proof**- Since the landmark is not changing its position in the environment, we can assume that
\[ n \Sigma_{k,i} = ^n \beta_{k-1} = 0. \] The lemma follows directly from equation (3.142).

\[
E[ ^n \alpha_k ] = ^n \alpha_{k-1} + [ A_k + ( R_k + ^n \Sigma_{k-1,i} ) \hat{Q}_k^{-1} ]^{-1} ( ^n \beta_{k-1} - ^n \alpha_{k-1} )
\]

(3.143)

\[
= ^n \alpha_{k-1} + [ A_k + ( R_k + 0 ) \hat{Q}_k^{-1} ]^{-1} ( 0 - ^n \alpha_{k-1} )
\]

(3.144)

\[
= ^n \alpha_{k-1} - [ A_k + ( R_k \hat{Q}_k^{-1} ) ]^{-1} ^n \alpha_{k-1}
\]

(3.145)

Therefore, observing the landmark at any time step concludes that \(^n \alpha_k\) is decreasing. The only exception arises when the error becomes zero. As a result, the expectation remains zero [8].

### 3.8.3 Lemma 3

If the absolute error on the position of the robot is smaller than the error in observation of the landmark (\(^n \alpha_k < ^n \beta_{k,i}\)) (in terms of its magnitude), \(^n \beta_{k,i}\) decreases in expectation as a result of the position error. If\(^n \alpha_k > ^n \beta_{k,i}\), the error in the position of the landmark may increase but the expectation will not exceed the error in estimating of the robot position [8].

**Proof** – The expected value on the position of the landmark is

\[
E[ ^n \beta_{k,i} ] = E[ h^{-1}( ^n x_k^R, z_{k,j} ) - m_{k,i} ] = E[ h^{-1}( ^n x_k^R, z_{k,j} ) ] - m_{k,i}
\]

(3.146)

\[
= E[ h^{-1}( ^n x_k^R, z_{k-1,j} ) + \Sigma_{k-1,i} ( R_k + \Sigma_{k-1,i} )^{-1} ( z_{k,j} - h^{-1}( ^n x_{k-1}^R, z_{k-1,j} ) + ^n x_{k-1}^R - u_k ) ] - m_{k,i}
\]

(3.147)
\[
\begin{align*}
    &= h^{-1}(n_{x_{k-1}}, n_{z_{k-1,j}}) + n \Sigma_{k-1,i} (R_k + n \Sigma_{k-1,i})^{-1} (E[n_{z_{k,j}}] - h^{-1}(n_{x_{k-1}}, n_{z_{k-1,j}}) + n_{x_{k-1}} - u_k) - m_{k,i} \\
    &= h^{-1}(n_{x_{k-1}}, n_{z_{k-1,j}}) + n \Sigma_{k-1,i} (R_k + n \Sigma_{k-1,i})^{-1} (n_{\alpha_{k-1}} - n_{\beta_{k-1}}) - m_{k,i} \\
    &= n_{\beta_{k-1,j}} + n \Sigma_{k-1,i} (R_k + n \Sigma_{k-1,i})^{-1} (n_{\alpha_{k-1}} - n_{\beta_{k-1}}) \\
    &= n_{\beta_{k-1,j}} + (A_k + (R_k (n \Sigma_{k-1,i})^{-1})^{-1} (n_{\alpha_{k-1}} - n_{\beta_{k-1}})
\end{align*}
\]

where \( h^{-1}(n_{x_{k-1}}, n_{z_{k-1,j}}) - m_{k,i} \) is the absolute error on the position landmark \( n_{\beta_{k-1,i}} \) at time \( k-1 \), and the last transformation indicates the linearity of the expectation. \( n \Sigma_{k-1,i} \) and \( R_k \) are both positive semi-definite, and therefore, the eigenvalues of \( (A_k + (R_k (n \Sigma_{k-1,i})^{-1})^{-1} \) are all positive and less than one [8].

### 3.8.4 Convergence

As explained in [8], it is assumed that \( n_{\hat{\beta}_{k}} \) is the maximum amount of variable \( n_{\beta_{k,i}} \) as the absolute error in location of landmark \( i \) at time step \( k \). \( n_{\hat{\beta}_{k}} \) can be expressed as

\[
    n_{\hat{\beta}_{k}} = \arg \max_{n_{\beta_{k,i}}} | n_{\beta_{k,i}} |
\]

Based on lemma 3, if the absolute robot position error \( n_{\alpha_{k-1}} \) exceeds the absolute error in location of landmark \( n_{\beta_{k,i}} \), there will be an increase in expectation of \( n_{\hat{\beta}_{k}} \). However, in the expectation, this will only be the case for a limited number of iterations, and Lemma 1
guarantees that $\alpha_{k-1}$ may only shrink in the expectation. Furthermore, Lemma 2 states that if at time step $k$ the landmark is observed, this error will shrink by a finite amount, regardless of the magnitude of $\beta_{k,j}$. As a result, $\alpha_{k-1}$ is smaller in magnitude in the expectation than the largest landmark error. Afterwards, Lemma 3 states that $\hat{\beta}_k$ will shrink in the expectation every time the landmark with largest error is observed. As a result, both errors $\hat{\beta}_k$ and $\alpha_k$ converge to zero since observing the landmark induces a finite reduction as stated in equation (3.90). To increase $\alpha_{k-1}$ to its old value in expectation, according to lemma 2, the total feature error must decrease in expectation which leads to an eternal decrease of the total feature error down to zero. Based on lemma 1, this error is an upper bound for the expected position error, and as a result, the expectation for the robot position error will converge. A corollary may be defined as; the filter converges in the expectation for the linear Gaussian SLAM only if all landmarks are observed infinitely, providing that the location of one landmark is known in advance [8].
3.9 Summary

Combined filters such as HybridSLAM remember long term uncertainty over robot’s path, due to the fact that the main covariance matrix of the system in EKF maintains covariance between pose and landmarks position for the rest of the calculation.

Combining EKF-SLAM and FastSLAM and representing it as Hybrid SLAM makes the filter overcome some issues where either filter becomes inconsistent. HybridSLAM is taking advantage of FastSLAM to deal with non-linearity of the system, estimating the path and mapping in local scale, and being less fragile to the data association problem. On the other hand, EKF-SLAM has the ability to recall uncertainty at each time step when needed. Since EKF-SLAM can deal with nonlinearity of the system, it may be an appropriate choice for a mixed filter. It should be noted though that EKF has its own estimation limitations and its approximation issues when facing a system with high nonlinearity. If a system is highly nonlinear, EKF as a sub-filter of HybridSLAM may not be able to represent a right global map and in the fusion step, the resulting map becomes inconsistent. An improved version of EKF called Unscented Kalman Filter (UKF), employs unscented transform in HybridSLAM and addresses the approximation issues in constructing a global map which results in a more reliable map in the fusion step.

The goal of this chapter is to replace UKF with EKF in HS and reformulate the algorithm. EKF approximates the state distribution using a Gaussian random variable and propagates it analytically through the first order linearization of a non-linear system. Therefore, EKF optimization may lead to suboptimal performance resulting in filter failure since it generates substantial error in the posterior mean and system covariance. UKF represents the state
distribution of the system based on a Gaussian random variable as well and a deterministic sample approach. In UKF based approaches however, a minimal set of sample points are chosen to calculate the mean and covariance of the Gaussian random variable. The samples are propagated through the high system non-linearity and more accurately approximate posterior belief of the robot to the second order Taylor series expansion. The approximation using UKF is more accurate compared to EKF speaking of which employs first order of Taylor series expansion specifically when motion and observation models are highly non-linear. In UKF, sigma points (minimal set of sample points) around the mean are used in a deterministic sampling technique called unscented Transform. By propagating sigma points through non-linear functions of motion and observation models, a high estimation of true mean and system covariance can be obtained. It is important to mention that unscented transform eliminates calculation of Jacobians in UKF by which the computation complexity is reduced substantially. However, the estimation is still based on a prediction and update manner. A local map is estimated by Rao-Blackwellised particle filter and fused to the global map previously estimated by UKF. The fusion process occur using Constrained Local Sub-map Fusion technique. CLSF compares the local map estimated at the vicinity of the robot and fuses the map into global map if necessary. Otherwise the global map remains as it is and the process repeats for the next step.
Chapter 4

Vehicle Model

4.1 Four Wheeled Non-holonomic Mobile Robot

In this chapter, kinematics methodologies for a four wheeled non-holonomic mobile robot will be presented. Kinematical model of a mobile robot may be derived in different means where some models may slow the system. Therefore, a careful derivation is necessary to prevent difficulties. First, a platform of a standard outdoor car-like robot based on its centre of gravity will be presented. Then, the platform is developed to provide a mathematical model particular to motion and sensing aspects. In the next sections, motion and observation models are linearized to fit into both UKF suboptimal filters instances in order to estimate path, local map, and global map. All simulation results will be based on equations and specifications of the model to fit into UKF filter.

4.1.1 Wheeled Mobile Robot System

Mobile robots need a driving system to give them mobility through an environment. In most cases, the locomotion system consists of wheels rather than legs. A driving system consisting of wheels give the robot more degree of freedom to freely move in the environment. Most wheeled autonomous mobile robots are car-like, consisting of four or six wheels. A driving system consisting of wheels leads a less complex procedure to derive motion equations and hence make the model suitable for navigational purposes.
4.1.2 A Car-Like Model

Figure 5.1 demonstrates a four wheeled non-holonomic mobile robot with two front wheels for steering and two rear wheels for driving purposes. To obtain a mathematical model for tracking control, there is a cumbersome calculation involved. However, a full development of the kinematics model is beyond the scope of this research, yet partially beneficial. It is fair to assume that the vehicle is given with a demanded velocity $v^d_k$ along with a steering angle $\alpha$ through the control vector $u_k$ at time step $k$.

Figure 4.1: A realistic car-like mobile robot
Subsequent to deriving a model with the above assumptions, a systematic noise will be added to note the error in the motion model. In figure 4.1, the vehicle is assumed with correct differential steering angles $\alpha_1$ and $\alpha_2$. In this study, a realistic model of a car-like robot with ICR intersection point of wheel axels is considered in order to eliminate slipping hypothesis when steering the front wheels. Rear wheels are assumed to have the differential mechanism to set longitudinal velocities of the rear wheels. $\alpha$ is the vehicle’s virtual steering angle located in the middle of the front axel. For wheels 1 and 2, coordinate systems are making angles $\alpha_1$ and $\alpha_2$, and change according to the steering function [67]. However, to obtain a model for simulation purposes $\alpha$ is considered as the steering angle resulted from a steering action, and $\theta$ is the angle between vehicle’s frame of reference and the global frame of reference.

### 4.2 Encoder Reading Translation

An optical encoder is attached to the robot’s wheel as shown in figure 4.2. By means of wheel encoders, the number of wheel rotations can be monitored and calculated. If the wheel diameter is known in advance, the displacement can be measured with ease. In the navigation process this displacement is referred as dead reckoning. The velocity measured by the encoder reader is denoted as $v^e_k$. This velocity is used to translate the velocity of the CG of the robot ($v^R_k$). The equation that is used for this translation purpose is described as

$$v^R_k = \frac{v^e_k}{1 - \tan(\alpha) \cdot \frac{W}{\ell}}$$  \hspace{1cm} (4.1)
4.3 Range/Bearing Sensor

The external range/bearing sensor is installed in front of the vehicle which is modeled by equation (A.6) in appendix A. A range/bearing sensor used for SLAM must be accurate enough to observe landmarks with minimum depletion, and a high accuracy depends on the function of the size, shape, and material type of the device reflector [11]. This sensor returns range and bearing information to landmark \( m_i \), then the \( i^{th} \) landmark position \( (z_{k,i} = (D, \beta)) \), is read. \( D \) is the distance of on-board sensor from the landmark and \( \beta \) is the sensor bearing measured with respect to coordinate system of the vehicle’s back axle \( (x^C_k, y^C_k) \). The steering angle of the vehicle is noted by \( \alpha \). In figures 4.2 and 4.3, all system parameters and variables are demonstrated.

\[ (x^C_k, y^C_k) \]

\[ \beta \]

\[ D \]

\[ \alpha \]

\[ \theta \]

Figure 4.2: The vehicle coordinate system and the position of \( i^{th} \) landmark with respect to the robot coordinate system.
4.4 Linearization to the First Order of Taylor Series

In practice, the prediction model for the robot trajectory and the model that relates an observation to the states are not linear. SLAM can be formulated properly if motion and observation models are presented in linear forms. For the linearization purpose, Jacobians of both models can be used to propagate the main covariance matrix of the system $P$ [10].

4.4.1 Motion Model Linearization

In this section, and prior to the linearization process, the model of the vehicle is obtained on the model in [11]. In these cases, usually the trajectory of one specific point of the robot body is analyzed [21]. In the robot shown in figure 5.2, the trajectory of the middle of the back axel is used to show the trajectory of center of the laser sensor and is illustrated by equation (4.2).

For clarity of the presentation, $X^R_k$ (path of the robot) is replaced by $X^S_k$ (path of the sensor). In fact, the state of the laser sensor mounted on the robot is the matter of importance, but to
simplify the signs and adapt it with the defined sets in chapter 1, character $S$ is replaced with character $R$. 

$$
\mathbf{v}_k^R = \begin{bmatrix}
\frac{\dot{x}_k^S}{x_k^R} \\
\frac{\dot{y}_k^S}{y_k^R} \\
\frac{\dot{\theta}_k^S}{\dot{\theta}_k^R}
\end{bmatrix} = \begin{bmatrix}
\frac{\dot{x}_k^S}{x_k^R} \\
\frac{\dot{y}_k^S}{y_k^R} \\
\frac{\dot{\theta}_k^S}{\dot{\theta}_k^R}
\end{bmatrix} = \begin{bmatrix}
v_k^R \cos(\theta) \\
v_k^R \sin(\theta) \\
v_k^R \tan(\alpha)
\end{bmatrix} + \mathbf{w}_k^R 
$$

(4.2)

$$
\mathbf{X}_k^S = \mathbf{X}_k^R = \{x_0^R, x_1^R, \ldots, x_k^R\} = \{\mathbf{X}_{k-1}^R, x_k^R\}
$$

(4.3)

The translation of the middle point of the back axle with respect to the global coordinate system $(^S_g x, ^S_g y)$ is representing the trajectory of the centre of the laser and is denoted as

$$
L^S = L^C + A \cdot \mathbf{t}_\theta + B \cdot \mathbf{t}_{\theta+\pi/2}
$$

(4.4)

where $L^S$ is the location of center of the laser sensor with respect to the global coordinate system, and $L^C$ is the location of the middle point of the robot’s back axle with respect to the global coordinate system. The transformation vector is defined by the orientation of the robot as

$$
\mathbf{t}_\theta = (\cos(\theta), \sin(\theta))
$$

(4.5)

The location of the sensor can be represented as following scalar equations:

$$
x_k^R = x_k^C + A \cdot \cos(\theta) + B \cdot \cos(\theta + \frac{\pi}{2})
$$

(4.6)
\[ y_k^R = y_k^C + A \cdot \sin(\theta) + B \cdot \sin(\theta + \frac{\pi}{2}) \] (4.7)

where \((x_k^S, y_k^S)\) is the laser sensor coordinate and \((x_k^R, y_k^R)\) is the robot coordinate with respect to the global coordinate system. In the MATLAB code related to simulations of this research in the next chapter \((x_k^S, y_k^S)\) will be considered as coordinates of the vehicle. The full state representation can be written as

\[
\begin{bmatrix}
\dot{x}_k^R \\
\dot{y}_k^R \\
\dot{\theta}_k^R
\end{bmatrix} =
\begin{bmatrix}
v_k^R \cdot \cos(\theta) - \frac{v_k^R}{\ell} \cdot N_1 \\
v_k^R \cdot \sin(\theta) + \frac{v_k^R}{\ell} \cdot N_2 \\
\frac{v_k^R}{\ell} \cdot \tan(\alpha)
\end{bmatrix} + w_k^R
\] (4.8)

where \(N_1 = (A \cos(\theta) + B \sin(\theta)) \cdot \tan(\alpha)\), and \(N_2 = (A \cos(\theta) - B \sin(\theta)) \cdot \tan(\alpha)\). The whole state can be modeled as

\[
\begin{bmatrix}
x_k^R \\
y_k^R \\
\theta_k^R
\end{bmatrix} =
\begin{bmatrix}
x_{k-1}^R + \Theta_{k-1}^z - \frac{v_{k-1}^R}{\ell} \Omega_{k-1}^x \\
y_{k-1}^R + \Theta_{k-1}^y - \frac{v_{k-1}^R}{\ell} \Omega_{k-1}^y \\
\frac{v_{k-1}^R}{\ell} \cdot \tan(\alpha_{k-1})
\end{bmatrix} + w_k^R
\] (4.9)

where

\[
\Theta_{k-1}^z = \Delta t \cdot v_{k-1}^R \cdot \cos(\theta_{k-1}^R)
\] (4.10)

\[
\Omega_{k-1}^x = (A \sin(\theta_{k-1}^R) + B \cos(\theta_{k-1}^R)) \cdot \tan(\alpha_{k-1})
\] (4.11)
\[
\Theta_{k-1}^2 = \Delta t \cdot v_{k-1}^R \cdot \sin(\theta_{k-1}^R) \quad (4.12)
\]

\[
\Omega_{k-1}^2 = \left( A \cos(\theta_{k-1}^R) - B \sin(\theta_{k-1}^R) \right) \cdot \tan(\alpha_{k-1}) \quad (4.13)
\]

\(w_k\) is the motion noise and \(\Delta t\) is the sampling time which in this case is not considered constant.

### 4.4.2 Sensor Model Linearization

The equation that relates the robot state to the sensor observations is

\[
\mathbf{z}_k = \mathbf{h}(\mathbf{x}) + \mathbf{v}_k = \begin{bmatrix} \mathbf{z}_{k,i}^D \mathbf{z}_{k,i}^R \end{bmatrix} + \mathbf{v}_k = \begin{bmatrix} \sqrt{(x_k^m - x_k^R)^2 + (y_k^m - y_k^R)^2} \\ \tan^{-1} \left( \frac{(y_k^m - y_k^R)}{(x_k^m - x_k^R)} \right) - \theta_k^R + \frac{\pi}{2} \end{bmatrix} + \mathbf{v}_k \quad (4.14)
\]

where \((x_k^m, y_k^m)\) is the coordinate of \(i^{th}\) landmark. Since landmarks are static, and at any time step \(k\), have their same locations with respect to the global coordinate system, the following equation can be defined.

\[
(x_{k+1}^m, y_{k+1}^m) = (x_k^m, y_k^m) = (x_i^m, y_i^m) \quad (4.15)
\]
4.4.3 Control System with Noise

If the state of the robot control system includes noisy signals (which in most cases it does), a noise vector $w^u_k$ is added to the control vector and the complete non-linear motion model can be expressed as

$$x_k = f(x_{k-1}, u_k + w^u_k) + w^R_k$$  \hspace{1cm} (4.16)

In most SLAM cases, the sum of control noise and noise of the state of the robot can be added. With Jacobian of control vector, the total noise of the motion is described as

$$w_k = J_u w^u_k + w^R_k$$ \hspace{1cm} (4.17)

where

$$J_u = \frac{\partial f(x, u)}{\partial u} \bigg|_{x=x_k, u=u_k}$$ \hspace{1cm} (4.18)

Now it can be seen that equation (4.19) is equivalent with equation (A.5) in appendix A.

$$x_k = f(x_{k-1}, u_k + w^u_k) + w^R_k \cong f(x_{k-1}, u_k) + w_k$$ \hspace{1cm} (4.19)
4.4.4 Noise Characteristics Matrices

All noise characteristics are assumed to be white, zero-mean, and independent.

\[ E[w_k] = E[w_k^R] = E[w_k^u] = E[v_k] = 0 \] (4.20)
\[ E[(w_k)(w_k)^T] = \delta_{i,j} \cdot Q_k \] (4.21)
\[ E[(w_k^R)(w_k^R)^T] = \delta_{i,j} \cdot Q_k^R \] (4.22)
\[ E[(w_k^u)(w_k^u)^T] = \delta_{i,j} \cdot Q_k^u \] (4.23)
\[ E[(v_k)(v_k)^T] = \delta_{i,j} \cdot R_k \] (4.24)

\[ \delta_{i,j} = \begin{cases} 
0 & i \neq j \\
1 & i = j 
\end{cases} \] (4.25)

\[ E[(w_k)(w_k)^T] = \delta_{i,j} Q_k = \delta_{i,j} (J_u Q_k^u J_u^T + \delta_{i,j} Q_k^R) \] (4.26)

where \( w_k \sim N(0, Q_k) \), \( v_k \sim N(0, R_k) \), \( w_k^R \sim N(0, Q_k^R) \), and \( w_k^u \sim N(0, Q_k^u) \). Now, equation (1.2) in chapter 1 can be rewritten in form of the following matrix:

\[ x_k = \begin{bmatrix} x_k^R \\ m \end{bmatrix} \] (4.27)

where

\[ x_k^R = (x_k^R, y_k^R, \theta_k^R)^T \in \mathbb{S}^3 \] (4.28)
\[ \mathbf{x}^m = (x_1^{m_1}, y_1^{m_1}, x_2^{m_2}, y_2^{m_2}, \ldots, x_u^{m_u}, y_u^{m_u}) \in \mathbb{S}^{2M} \] (4.29)

and the model can be written in the form of

\[ \mathbf{x}_k^R = f(\mathbf{x}_{k-1}^R, \mathbf{u}_k) + \mathbf{w}_k \] (4.30)

\[ \mathbf{x}^m_\alpha = \begin{bmatrix} x_\alpha^{m_1} \\ y_\alpha^{m_1} \end{bmatrix} \] (4.31)

### 4.4.5 Jacobian Matrices

The Jacobian matrix for the function of extended system \( f(\mathbf{x}_{k-1}, \mathbf{u}_k) \) is defined as

\[ \Phi_k = \left. \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x} = \mathbf{x}_{k-1}} = \begin{bmatrix} \frac{\partial f(\mathbf{x}^R)}{\partial \mathbf{x}^R} |_{\mathbf{x}^R = \mathbf{x}_{k-1}^R} & 0 \\ 0 & I \end{bmatrix} \] (4.32)

where \( I \in \mathbb{S}^{2M} \times \mathbb{S}^{2M} \) and the Jacobian matrix of the observation function \( h(\mathbf{x}_k) \) is

\[ \Psi_k = \left. \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x} = \mathbf{x}_k} = \begin{bmatrix} \frac{\partial z_i^\beta}{\partial z_i^\beta} \\ \frac{\partial z_i^\beta}{\partial \mathbf{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial z_i^\beta}{\partial (\mathbf{x}_k^R, \mathbf{m})} \\ \frac{\partial (\mathbf{x}_k^R, \mathbf{m})}{\partial \mathbf{x}} \end{bmatrix} \] (4.33)
Equation (4.33) always has a large number of null elements since only a few landmarks are observed and validated by the sensor at each time step. For instance, if only one landmark is observed, the Jacobian becomes:

$$
\psi_k = \begin{bmatrix}
\Delta x & \Delta y & 0 & 0 & \cdots & -\Delta x & -\Delta y & 0 & \cdots & 0 \\
\Delta y & \Delta x & 0 & 0 & \cdots & -\Delta y & -\Delta x & 0 & \cdots & 0 \\
-\Delta y & -\Delta x & -1 & 0 & \cdots & \Delta y & \Delta x & -1 & 0 & \cdots & 0 \\
-\Delta x & -\Delta y & \Delta x & \Delta y & \cdots & \Delta x & \Delta y & \cdots & \Delta x & \Delta y & \cdots & 0 \\
\Delta x & \Delta y & \cdots & \Delta x & \Delta y & \cdots & \Delta x & \Delta y & \cdots & 0 & \cdots & 0 \\
\end{bmatrix}
$$

(4.34)

where

$$
\Delta x = x^R_k - x^m_k
$$

(4.35)

$$
\Delta y = y^R_k - y^m_k
$$

(4.36)

$$
\Delta = \sqrt{(\Delta x)^2 + (\Delta y)^2}
$$

(4.37)

### 4.5 Second Order Linearization Using Sigma-Point

In appendix A, UKF is briefly discussed with an expression of its sigma-point based algorithm in section A.1.3.4. Furthermore, it was explained how a system function may be expressed as a Taylor series in a polynomial form in chapter 3 section 3.3.2. A major shortcoming of EKF is that the filter is limited to the first order accuracy of the propagated mean and covariance as a result of first order truncated Taylor series linearization technique, similar to what was discussed in section 4.4. UKF can address this issue through the use of a deterministic sampling approach to approximate the optimal gain and prediction terms in Gaussian approximate linear Bayesian framework. The UKF approach is quite different from that of EKF implementation on a highly non-linear system. UKF, with its non-derivative Gaussian random variable propagation technique, handles a different implementation of a
general deterministic sampling framework for the calculation of the posterior mean and covariance of the system to the second order of Taylor series. To show how the model robot can be approximated as part of a non-linear system in the second order of a Taylor series (discussed in section 4.1.2), a non-derivative technique based on Sterling’s polynomial interpolation formula [68] is used for this particular model.

### 4.5.1 Second Order Sterling Polynomial Interpolation

The formulation of the second order Sterling Polynomial Interpolation (SPI) is derivation of the Divided Deference Filter (DDF) [69] and the Central Difference Filter (CDF) [70] and on the basis work of [68]. To formulate the equations of the system in a linear form, the second order SPI will be briefly discussed in this section to illustrate how a non-linear system can be approximated in a linear form. Subsequently, system mean and covariance in the posterior state will be discussed. Based on the equation (3.20) in section 3.3.2, a non-linear function of a random variable $x$ around a statistical point $\bar{x}$ as its mean, can be expressed by

$$h(x) = h(\bar{x}) + D_{\delta_x} h + \frac{1}{2!} D_{\delta_x}^2 h + \cdots = h(\bar{x}) + (x - \bar{x}) \frac{dh(x)}{dx} + \frac{1}{2!} (x - \bar{x})^2 \frac{d^2 h(x)}{dx^2}$$

(4.38)

The SPI formula [71] utilizes a finite number of functional evaluations to approximate the above non-linear function with $\tilde{D}_{\lambda_x}$ as the first, and $\tilde{D}_{\lambda_x}^2$ as the second order central divided difference operators acting on $h(x)$. $\ell$ is the interval length or central difference step size and $\bar{x}$ is the prior mean of $x$ around which the expansion is done. The resulting formula can be expressed as
\[ h(x) = h(\bar{x}) + \tilde{D}_{\Delta x} h + \frac{1}{2!} \tilde{D}^2_{\Delta x} h \]  \hfill (4.39) \\

\[ \tilde{D}_{\Delta x} = (x - \bar{x}) \frac{h(\bar{x} + \ell) - h(\bar{x} - \ell)}{2\ell} \]  \hfill (4.40) \\

\[ \tilde{D}^2_{\Delta x} = (x - \bar{x})^2 \frac{h(\bar{x} + \ell) + h(\bar{x} - \ell) - 2h(\bar{x})}{\ell^2} \]  \hfill (4.41) \\

In the case where analytical derivatives are replaced by central divided differences, the SPI formula can be interpreted as the Taylor series discussed in section 3.3.2. If this formula is extended to the multi dimensional case, the function \( h(x) \) may be obtained by “first stochastically decoupling” [72] of the prior random variable \( x \) through the following linear transformation.

\[ y = S_x^{-1} x \]  \hfill (4.42) \\

\[ \tilde{h}(y) = h(S_x y) = h(x) \]  \hfill (4.43) \\

where \( S_x \) is called Cholesky factor (CF) of the covariance matrix \( P_x \) of \( x \) such that \( P_x = S_x S_x^T \) [73]. It should be noted that Taylor series expansion of \( h(.) \) is identical with \( \tilde{h}(.) \), if the expected value of vector \( x \) is \( E[x] \) and the covariance of the system is \( P_x = E[(x - \bar{x})(x - \bar{x})^T] \). The transformation stochastically decouples variables in \( x \) so that \( y \) interval components become mutually uncorrelated. So the covariance of \( y \) can be expressed as

\[ P_y = E[(y - \bar{y})(y - \bar{y})^T] = I \]  \hfill (4.44)
assuming that \( L \) is the dimension of \( \mathbf{x} \) and \( \mathbf{y} \) with \( \Delta_{y_i} = (y - \bar{y})_i \) as the \( i^{th} \) component of \( y - \bar{y} \) (\( i = 1, \ldots, L \)), \( e_i \) is the \( i^{th} \) unit vector, \( d_i \) is the partial first order difference, \( d_i^2 \) is the partial second order difference, and \( m_i \) is the mean operator [74]. Components of \( \tilde{h}(\mathbf{y}) \) can be expressed as

\[
\tilde{D}_{\Delta \mathbf{y}} \tilde{h} = \left( \sum_{i=1}^{L} \Delta_{y_i} m_i d_i \right) \tilde{h}(\bar{y})
\]  
(4.45)

\[
\tilde{D}_{\Delta \mathbf{y}}^2 \tilde{h} = \left( \sum_{i=1}^{L} \Delta_{y_i}^2 d_i^2 + \sum_{j=1}^{L} \sum_{q=1}^{L} \Delta_{y_j} \Delta_{y_q} (m_j d_j^2)(m_q d_q) \right) \tilde{h}(\bar{y})
\]  
(4.46)

where

\[
d_i \tilde{h}(\bar{y}) = \frac{1}{2\ell} \left[ \tilde{h}(\bar{y} + \ell e_i) - \tilde{h}(\bar{y} - \ell e_i) \right]
\]  
(4.47)

\[
d_i^2 \tilde{h}(\bar{y}) = \frac{1}{2\ell^2} \left[ \tilde{h}(\bar{y} + \ell e_i) + \tilde{h}(\bar{y} - \ell e_i) - 2\tilde{h}(\bar{y}) \right]
\]  
(4.48)

\[
m_i \tilde{h}(\bar{y}) = \frac{1}{2} \left[ \tilde{h}(\bar{y} + \ell e_i) + \tilde{h}(\bar{y} - \ell e_i) \right]
\]  
(4.49)

Using equations (4.42) and (4.43) and considering that \( s_{x_i} \) is the \( i^{th} \) column of the Cholesky factor of covariance matrix of \( \mathbf{x} \) can be induced as

\[
\tilde{h}(\bar{y} \pm \ell e_i) = h(S_x [\bar{y} \pm \ell e_i]) = h(S_x \bar{y} \pm \ell S_x e_i) = h(\bar{x} \pm \ell s_{x_i})
\]  
(4.50)

\[
s_{x_i} = S_x e_i = (S_x)_{i,} = (\sqrt{P_x})_{i}
\]  
(4.51)
Set of vectors defined in equation (5.50) is equivalent in form in the same way that the UKF generates its set of sigma-points in equations (3.19) to (3.20) in section 3.1.3.4, with only the difference being in the value of the weighting term.

### 4.5.2 Posterior Mean and Covariance Estimation

The observation function can be expressed through a non-linear function \( h(.) \) and with considering non-linear transformation of an \( L \) dimensional random variable \( x \) (the motion vector) with covariance \( P_x \) and mean \( \bar{x} \) as

\[
\mathbf{z}_k = h(\mathbf{x}_k) = \tilde{h}(\mathbf{y}_k) \approx \tilde{h}(\mathbf{y}_k) + \tilde{D}_{\Delta y} \tilde{h} + \frac{1}{2} \tilde{D}_{\Delta y}^2 \tilde{h} \\
y = S_x x
\]

The posterior mean of \( y \) and its covariance and cross covariance are defined as

\[
\bar{z}_k = \mathbb{E}[\mathbf{z}_k] \\
P_{z_k} = \mathbb{E}[ (\mathbf{z}_k - \bar{z}_k) (\mathbf{z}_k - \bar{z}_k)^T ] \\
P_{x_k z_k} = \mathbb{E}[ (\mathbf{x}_k - \bar{x}_k) (\mathbf{z}_k - \bar{z}_k)^T ]
\]

Assuming that \( \Delta_y = (y - \bar{y}) \) is a zero-mean unity variance random variable which is symmetric as defined in equation (4.42), the mean is approximated as
\[ \bar{z}_k \approx E[\tilde{h}(y_k)] + \tilde{D}_h \tilde{h} + \frac{1}{2} \tilde{D}_h^2 \tilde{h} = \tilde{h}(y_k) + E[\frac{1}{2} \tilde{D}_h^2 \tilde{h}] \] (4.57)

\[ = \tilde{h}(y_k) + E[\frac{1}{2\ell^2} \left( \sum_{i=1}^{L} \Delta_{y_i}^2 d_i^2 \right) \tilde{h}(y_k)] \] (4.58)

\[ = \tilde{h}(y_k) + \frac{1}{2\ell^2} \sum_{i=1}^{L} \left[ \tilde{h}(y_k + \ell e_i) + \tilde{h}(y_k - \ell e_i) - 2\tilde{h}(y_k) \right] \] (4.59)

\[ \frac{\ell^2 - L}{\ell^2} \tilde{h}(y_k) + \frac{1}{2\ell^2} \sum_{i=1}^{L} \left[ \tilde{h}(\bar{y}_k + \ell e_i) + \tilde{h}(\bar{y}_k - \ell e_i) \right] \] (4.60)

Rewriting the posterior mean in terms of motion vector results in

\[ \bar{z}_k = \frac{\ell^2 - L}{\ell^2} \tilde{h}(y_k) + \frac{1}{2\ell^2} \sum_{i=1}^{L} \tilde{h}(\bar{y}_k + \ell s_{x_i}) + \tilde{h}(\bar{y}_k - \ell s_{x_i}) \] (4.61)

using the identity

\[ \bar{z}_k = E[z_k] = E[z_k] + h(\bar{x}_k) - h(\bar{x}_k) = E[z_k] + h(\bar{x}_k) - E[h(\bar{x}_k)] \] (4.62)

\[ = h(\bar{x}_k) + E[z_k - h(\bar{x}_k)] \] (4.63)

the covariance can be obtained as

\[ P_{zk} = E[(z_k - \bar{z}_k)(z_k - \bar{z}_k)^T] \] (4.64)

\[ = E[(z_k - h(x_k))(z_k - h(x_k))^T] - E[(z_k - h(x_k))] E[(z_k - h(x_k))^T] \] (4.65)

\[ = E[(z_k - \tilde{h}(y_k))(z_k - \tilde{h}(y_k))^T] - E[(z_k - \tilde{h}(y_k))] E[(z_k - \tilde{h}(y_k))^T] \] (4.66)
From equation (4.52), the second order approximation of $z_k - \tilde{h}(y_k) = \tilde{D}_{\Delta_y} \tilde{h} + \frac{1}{2} \tilde{D}_{\Delta_y}^2 \tilde{h}$ can be substituted into equation (4.66) and the posterior can be achieved as follows

$$P_{z_k} \approx E[(\tilde{D}_{\Delta_y} \tilde{h} + \frac{1}{2} \tilde{D}_{\Delta_y}^2 \tilde{h}) (\tilde{D}_{\Delta_y} \tilde{h} + \frac{1}{2} \tilde{D}_{\Delta_y}^2 \tilde{h})^T]$$

$$- E[(\tilde{D}_{\Delta_y} \tilde{h} + \frac{1}{2} \tilde{D}_{\Delta_y}^2 \tilde{h}) E[(\tilde{D}_{\Delta_y} \tilde{h} + \frac{1}{2} \tilde{D}_{\Delta_y}^2 \tilde{h})]^T]$$

(4.67)

Since $\Delta_y = (y - \bar{y})$ is symmetric, all resulting odd-order expected moments have zero value.

To keep the results computationally tractable, all components of the resulting fourth order term, $E[\frac{1}{4} (\tilde{D}_{\Delta_y}^2 \tilde{h}) (\tilde{D}_{\Delta_y}^2 \tilde{h})^T]$, that contains cross differences in the expansion of equation (4.67) will be discarded. The reason is that the inclusion of these terms leads to an excessive increase in the number of computations as the number of such terms rapidly grows with the dimension of $y$. The extra effort is not considered since it is not possible to capture all fourth order moments [68]. The covariance approximation and cross-covariance matrices are expressed as below. For details on how to derive these approximations, see [73].

$$P_{z_k} \approx \frac{1}{4\ell^2} \sum_{i=1}^{L} [h(\bar{x}_k + \ell s_{x_i}) - h(\bar{x}_k)] [h(\bar{x}_k + \ell s_{x_i}) - h(\bar{x}_k)]^T$$

$$+ \frac{\ell^2 - 1}{4\ell^3} \sum_{i=1}^{L} [h(\bar{x}_k + \ell s_{x_i}) + h(\bar{x}_k - \ell s_{x_i}) - 2h(\bar{x}_k)]$$

$$\times [h(\bar{x}_k + \ell s_{x_i}) + h(\bar{x}_k - \ell s_{x_i}) - 2h(\bar{x}_k)]^T$$

(4.68)
\[
P_{x_kz_k} = E[(x_k - \bar{x}_k) (z_k - \bar{z}_k)^T]
\]
\[
\approx E[(S_x (y_k - \bar{y}_k) [\tilde{D}_{\Delta_y} \tilde{h} + \frac{1}{2} \tilde{D}_{\Delta_y}^2 \tilde{h}] - E[\frac{1}{2} \tilde{D}_{\Delta_y}^2 \tilde{h}])^T]
\]
\[
= E[(S_x (y_k - \bar{y}_k) [\tilde{D}_{\Delta_y} \tilde{h}]^T] + \frac{1}{2} E[(S_x (y_k - \bar{y}_k) [\tilde{D}_{\Delta_y}^2 \tilde{h}]^T]
\]
\[
- \frac{1}{2} E[(S_x (y_k - \bar{y}_k)] \times E[\frac{1}{2} \tilde{D}_{\Delta_y}^2 \tilde{h}]^2
\]
\[
= E[(S_x (y_k - \bar{y}_k) [\tilde{D}_{\Delta_y} \tilde{h}]^T]
\]
\[
= \frac{1}{2\ell} \sum_{i=1}^{\ell} s_{x_i} [h(\bar{y}_k + \ell e_i) - \tilde{h}(\bar{y}_k - \ell e_i)]^T
\]
\[
= \frac{1}{2\ell} \sum_{i=1}^{\ell} s_{x_i} [h(x_k + \ell s_{x_i}) - h(x_k - \ell s_{x_i})]^T
\]

In equation (4.70) the odd-order moment terms are all zero since the term \((y_k - \bar{y}_k)\) is symmetric. The optimal setting of the central difference interval parameter \(\ell\) is dictated by the prior distribution of \(y = S_{x}^{-1}x\). Therefore, \(\ell^2\) has to be equal to the kurtosis of \(y\) to minimize errors between the true mean and covariance and their estimates. For Gaussian priors, the optimal value of \(h\) is thus \(h=\sqrt{3}\). For further detail refer to [68]. To approximate the posterior statistics of the system state in the presence of observation noise \(v_k\) and motion noise \(w_k\), the Central Difference Kalman Filter (CDKF) as an extension of sigma-point Kalman filter is used [76]. It is assumed that

\[
R_k = E[(v_k)(v_k)^T]
\]  
(4.74)

\[
Q_k = E[(w_k)(w_k)^T]
\]  
(4.75)
where $R_k$ is the covariance matrix of observation noise and $Q_k$ is the covariance matrix of motion noise, and both are considered as Gaussian distribution functions. A complete algorithm for the linearization to the second order of the Taylor series for the car-like robot similar to figure 5.1 can be found in appendix D.
4.6 Summary

A realistic outdoor mobile robot model was shown in this chapter. The car like mobile robot has a mounted range/bearing sensor to detect landmarks and a control system equipped with optical encoder attached to its wheel. The optical encoder monitors robot’s motion. Motion and sensor modeling procedure of such a robot is a necessary step to prepare the robot for real world applications. It was also assumed that the corrupting noise in both sensor and motion models are Gaussian, with zero-mean, independent and white.

Derived models must be transformed to linear forms since the real dynamic systems are mostly non-linear. Usually, wheeled vehicles are good candidates for this purpose since their motion models are suitable for linearization. Jacobian matrices are used to linearize both motion and sensor models. Once motion and sensor models are linearized, they can be fit into the algorithm to allow it to make the best estimation of robot’s path. If the motion is highly non-linear, using Taylor series approximation would not be sufficient. Since the UKF algorithm is used as a sub-filter for the proposed algorithm, a linearization method based on the second order Sterling Polynomial Interpolation is employed to linearize the highly non-linear motion through the second term of Taylor series. The second order Sterling Polynomial Interpolation is using a finite number of functional evaluations to approximate a non-linear function with first and second order central divided difference operators. These operators act on the observation function expressed in a non-linear form considering non-linear transformation of an L dimensional random variable.
Chapter 5

Results and Discussion

Throughout this chapter, different scenarios are simulated to explore and validate characteristic features of Unscented HybridSLAM algorithm. The simulations invoke a linearized motion model as well as an observation model that provides relative data (from range/bearing sensor) for the filter calculations. At the same time, the performance of the proposed algorithm will be thoroughly compared with currently used filters such as EKF-SLAM, FastSLAM, and HybridSLAM. Figure 5.1 demonstrates vehicle dimensions and specifications considered for this study. According to this figure, $A=4.0\text{m}$, $\ell=3.0\text{m}$, $W=0.75\text{m}$, and $B=0.4\text{m}$. The average velocity is $2\ \text{m/s}$ and the maximum range for the mounted sensor is $100.00\ \text{m}$. In this simulation, the noise in which the system is subject to is assumed Gaussian, white with zero-mean, and independent. First, a maximum of $500\text{m by 500m}$ and minimum of $50\text{m by 50m}$ environment is considered as a whole global map. Then a simulation according to this assumption is made using a processor in appendix E. It is assumed that the robot’s true initial location in the environment is given a priori. This is the very first version of the true map of the environment in which the location of the robot at time step zero is initialized. It is also assumed that uncertainties sourced from observation and motion, are incorporated into the calculation. It should be noted that uncertainty in motion is independent from uncertainty in observation which happens to be a fair assumption for real applications as well. In the next sections, simulations will start by demonstrating the performance of currently used algorithms, of primary interest, FastSLAM, EKF-SLAM, and HybridSLAM. Next, simulations will be repeated using Unscented HybridSLAM for the same
scenario and compared to the above filters. Different aspects of the algorithm will be thoroughly investigated and discussed on the simulated data.

![Figure 5.1: Specifications of the outdoor mobile robot [8]](image)

### 5.1 Performance Comparison

Prior to evaluating UHS filter, a SLAM scenario will be simulated using the gold standard algorithm EKF-SLAM in order to compare other methods with EKF as the gold standard solution and in particular to find the discrepancies in performances and optimality of the proposed filter with other filters. Equations of motion and observation models are linearized and both are presented to the algorithm in the form of Gaussian. Noise is assumed as Gaussian, white, and with zero-mean. Motion noise and observation noise are independent. A simulation according to these assumptions is made using EKF algorithm under Gaussian conditions. The Root Mean Square (RMS) position error and Orientation Error (OE) are demonstrated and discussed in each case. Furthermore, through a step by step simulation, disadvantages of each filter is thoroughly investigated and discussed.
5.1.1 Gaussian Implication

The true path (blue-dashed line), estimated path performed by EKF-SLAM (red-solid line), and the odometric readings of the robot (black-dashed line) are demonstrated in Figure 5.2-a. In figure 5.2-d, the robot remains near the true path with a high accuracy. Figures 5.2-b, 5.2-e, and 5.2-f, illustrate EKF-SLAM performance precision when the Gaussian Condition is implied. In figure 5.2-c, observation of a landmark is projected in a short period of time in which the accuracy in the landmark location estimation is high. Figure 5.3 illustrates similar trajectory for the same robot using the same linearized equations. This time, the noise is not Gaussian but is a mixture of exponential and Gaussian. Results show how fragile the EKF-SLAM is when it is disturbed by non-Gaussian noise. Figure 5.3-c illustrates the estimated location of a landmark with its true location at (x=350m, y=150m) which demonstrates that for this particular scenario, the landmark has most likely been observed around (x=352m, y=154m) causing the RMS position error to be offset about 10m indicating that EKF is not able to manage the non-Gaussian error of the system. Figures 5.3-e and 5.3-f demonstrate the difference along both x and y-axes that are made after the robot arrives to the end of the trajectory. Another reason for this failure could be contributed to the fact that EKF can not deal with the non-linearity of the observations. If for some reason, the observation information is wrong, it will be embedded to the filter before fusing the data to the map with no more reconsideration. Since EKF algorithm does not perform any correction at this step, the wrong data will become part of the map and cannot be corrected. In the event that more wrong observations are obtained in the next time steps, the error becomes enormous and the EKF diverges. Comparing observations in figures 5.2-c and 5.3-c indicate that observation of landmarks under non-Gaussian conditions make the filter diverge at some point [41].
Figure 5.2: Applying EKF-SLAM under Gaussian Conditions and linearized motion and observation models. a) Comparing true path and estimated path b) Orientation error c) observation of landmarks from a close point of view d) RMS position error e) error in x direction f) error in y direction [41]
Figure 5.3: Applying EKF-SLAM under Non-Gaussian Conditions, motion and observation models are still linearized. a) Comparing true path and estimated path b) Orientation error c) observation of landmarks from a close point of view d) RMS position error e) error in x direction f) error in y direction [41]
5.1.2 Nearby Landmarks

When confronting nearby landmarks, the single hypothesis data association property of EKF-SLAM produces a substantial error on the localization and mapping process and due to growing number of elements in the main covariance matrix, EKF is not able to deal with more than a few hundred landmarks. The main system covariance matrix grows quadratic as more landmarks are detected by the range/bearing sensor and the correlation between each pair of landmarks in the map becomes stronger. What finally remains is a solid map as a result of erroneous readings, meaning that there are too many wrong landmarks that are observed. Moreover, EKF algorithm is based on single data association hypothesis and consequently, cannot handle the uncertainty of observing nearby landmarks. In such a case, FastSLAM is able to do the estimation task, however, it should be noted that there will be a need for a large number of particles. For nearby landmark scenarios, there will be lots of particles with low weights, and as a result, withdrawn from the calculation in the re-sampling process. If there are enough particles for a large trajectory, the estimation process may be done successfully, but the process will be time consuming and costly. There will be simulations in next sections to demonstrate how number of particles may affect computation costs and process time.

In figure 5.4 the EKF-SLAM is used for nearby landmarks. Results indicate that the filter fails to do the estimation task. Figure 5.5 illustrates the same situation using RBPF where the non-Gaussian assumption is considered as well. Nonetheless, the filter is performing well in estimating the path and landmarks positions. It is important to note that this high level performance is the result of using 1000 particles. For the robot position tracking, 1000 particles are considered as well. The results confirm the outperformance of RBPF over EKF with regard to a situation in which the system noise is not Gaussian and at the same time the
motion and observation models are non-linear. The results demonstrate how well the RBPF deals with a situation in which EKF could not, except for the fact that the process time has increased substantially. Once the number of particles is reduced, the process time decreases but the RBPF performance lowers as well. The fact is that increasing the number of particles will increase the time of re-sampling step when the particles with low weight are eliminated from the rest of the process [76].

Figure 5.4: Applying EKF-SLAM for nearby landmarks with linear motion and observation models, the noise is Gaussian. a) Comparing true path and estimated path b) Orientation error c) error in x direction d) error in y direction [76]
Figure 5.5: Applying FastSLAM for nearby landmarks with non-linear motion and observation models, the noise is not Gaussian and there are 1000 particles used for the sampling process. a) Comparing true path and estimated path b) Orientation error c) error in x direction d) error in y direction [76]

Figure 5.6 demonstrates the RMS position error for two different numbers of particles used in the same scenario. As the number of particles reduce from 1000 to 100, the process time decreases. However, the average RMS position error increases from 0.3m for 1000 particles to 0.5m for 100 particles. The process time is less than 1400 seconds for 100 particles and 2350 seconds if 1000 particles are used. While using 100 particles reduces the process time substantially, it does not give the same promising results when 1000 particles are used [76].
To compare the performance of the proposed filter, RMS position error and orientation error are calculated for the same scenario; the motion is highly non-linear, the motion noise is non-Gaussian, and there are nearby landmarks in a long trajectory corridor-like path. Simulation results in figure 5.7 show that the RMS position error using 100 particles in UHS has the same promising results if 1000 particles are used in FastSLAM. While HS is expected to have the same results as UHS, one should consider that the motion is highly non-linear and the furthest linearization is to the first order of Taylor series as described in chapter 5 section 5.4. Consequently, the EKF-SLAM sub-filter in HS will be estimating the global map with a low accuracy. The main difference between HS and UHS is the non-linearity of the system, and while HS has a shortcoming in handling a high non-linearity of the motion, Simulation results indicate that UHS is quite consistent. Figure 5.8-a compares UHS with three different filters in terms of RMS position error for 100 iterations. The number of particles used for UHS, HS, and FastSLAM is 100 in presence of 1800 nearby landmarks lined up to both sides of the path. The distance between mutual landmarks is approximately 5m. EKF-SLAM would not
even be close to the desired performance due to its sensitivity to motion non-linearity as well as high ambiguity of data associated with location of landmarks [76].

For a comparison among all mentioned filters, 100 iterations of the above scenario were simulated for each filter. Figure 5.8-b compares the Orientation error for each specific filter. Results show that unless a large number of particles are used, UHS outperforms FastSLAM and HS. If a large number of particles are used, FastSLAM will have the most accurate estimation of the location, however, it should be noted that the process time will be substantially increased. The normal process time using 100 particles for FastSLAM is around 1400s. Just by increasing number of particles 10 times, the process time increases up to 2300s. However, HS and UHS have more reasonable computation time. The process time for both UHS and HS is close to 2000s recalling that UHS is outperforming HS. Through iterations for different filters, it will be shown in next sections that the constant difference of process time highly depends on number of observations that are incorporated into the calculation for each time step [44].
Figure 5.8: Comparison of different filters performances in terms of a) RMS position error  b) Orientation error [44]
5.1.3 Observation and Motion Noise

In this section, UHS performance based on SLAM characteristic features is investigated and simulated results are compared to FastSLAM and the original HS for a new scenario. All robotic vehicle specifications remain the same as before but at this instance, the maximum range for the range/bearing sensor is set to 200m. Such a range for a laser scanner would be unrealistic in real applications. However, the goal in this section is to evaluate the computational abilities of Unscented HybridSLAM. Therefore, such range setting (200m) does not really affect the evaluation process. In a simulated scenario the robot is traveling on a road sided by nearby landmarks with minimum 0.10m distant. In these simulations the sensor is supposed to detect approximately 12 point landmarks per meter. In all simulations shown in figures 5.9 to 5.12, motion noise is assumed zero and observation noise is 0.10m. The difference between observation noise and motion noise is considered intentionally high (0.1m) to investigate the performance of different algorithms.

![Figure 5.9: Robot path and waypoints [76]](image_url)
Figure 5.9 shows the true path, nearby landmarks, and waypoints. Figure 5.10 depicts the performance of each particular filter in terms of root mean square position error. Figure 5.10-a indicates that due to sample impoverishment in the re-sampling step, FastSLAM algorithm (with 200 particles) does not manage the navigation task properly. In Figure 5.10-b, the performance of FastSLAM algorithm is quite acceptable, however, due to usage of a large amount of particles, the sample impoverishment process is quite slow and the time consumed to fulfill the navigation process is substantially large. Figure 5.10-c depicts the same situation using HS method and 200 particles as well. In this figure, HS does the estimation process with quite more accuracy compared to FastSLAM in figure 5.10-a. The combination of EKF-SLAM and FastSLAM in HS, performs the navigation task with average RMS position error of 0.80m, even though the number of used particles are the same as the FastSLAM case in figure 5.10-a. To demonstrate the UHS performance, the same scenario was repeated. With UKF management abilities in UHS, RMS position error is reduced substantially. Instead of estimating the global map by EKF in HS, the estimation is taken over by UKF. When estimating the global map, UKF is employing sample points that capture the true mean and covariance of the system. This accuracy is the result of sample point propagation through the true system non-linearity by including the second order of Taylor series expansion. The estimation process time is however, longer than usual. Nonetheless, the estimation time difference on the same path between HS and UHS is not much and can be neglected in a long trajectory. The main reason for such a difference could be due to a more calculation complexity in UHS compared to HS. The HS and EKF employ the first order and in UHS, UKF employs the second order of Taylor series expansion. Both HS and UHS share the same linear complexity as well as the same low sensitivity to data association problem [77].
Figure 5.10: RMS position error for a) FastSLAM using 200 particles b) FastSLAM using 5000 particles c) HybridSLAM using 200 particles d) Unscented HybridSLAM using 200 particles [76]
Figure 5.11: Map produced by FastSLAM a) 200 particles b) 5000 particles [78]
Figure 5.12: Map produced by a) HS b) UHS [78]
Maps produced by FastSLAM in the same scenario are shown in figure 5.11 where by using 5000 particles a highly accurate map is resulted. Figure 5.12 demonstrates maps produced by HS and UHS filters. In these graphs, both filters have managed to make a map of nearby landmarks accurately. A comparison between map produced by HS and UHS using 200 particles indicates that both are out-performing FastSLAM while at the same time, the map produced by UHS is slightly more accurate than the one produced by FastSLAM using 200 particles. If either observation or motion noise is close to zero, the performance of FastSLAM usually lowers. While this accuracy of noise could be an advantage in a standard filter such as EKF-SLAM, it makes RBPF to fail in FastSLAM. In the event when the motion in the system is noisy but the measurement noise is zero, the filter starts to fail. This is an important disadvantage of a standard algorithm based on particle filter. If the sensor is very accurate which means its error is close to zero, and at the same time the motion is very noisy, the observation noise and the robot noise will not match. The reason for this is contributed to the fact that many particles will be thrown in the re-sampling step and there is not enough particles left to be incorporated for the rest of the path estimation. Consequently, the filter diverges and the result will be catastrophic. This property that affects the performance of FastSLAM is referred as “Sample Impoverishment”. In all the following situations, the linear velocity noise is 0.1 and the observation noise is zero. Plots for RMS position error and orientation error indicate that they increase as the observation noise is assumed zero for FastSLAM algorithm. It is important to mention that this situation occurs only when the proposal and posterior distribution are mismatched which can be the result of a big difference between motion noise and observation noise. The motion model spreads the particles out over a large space and only a small fraction of particles receive non-negligible weights. As a result,
there will not be many particles left to estimate the path at the next time step. Furthermore, as more landmarks are observed, many more particles are thrown out and the problem becomes more compounded. There is one solution to overcome this problem where by increasing number of particles, statistical samples will last more. This, however, leads to more calculation and consequently more memory and is rather time consuming. Moreover, the filter becomes over confident and particles which carry uncertainty over the path, will be ignored and consequently, after a long course, the algorithm loses its sense of uncertainty. Losing this sense makes the algorithm overconfident and it will ultimately fail [78].

5.1.4 Process Time

In Figure 5.13 performance of three filters, FastSLAM, HS, and UHS are compared in the same experiment in the previous section. 432 observable landmarks are incorporated into each filter calculation. Different number of particles was used to evaluate the performance of each filter and each experiment was run three times for 20 different numbers of particles from 1 to 5000 with an increase of 100 particles at each experiment. RMS position error was measured versus the number of particles used in different filters. While in all cases, at some point, decreasing number of particles results in an exponential growth in the RMS position error, the increase happens at different number of particles in each case. For FastSLAM, the exponential growth shows around 1000 particles. RMS position error growth in HS is relatively close to FastSLAM and starts around 700 particles, while the increase in UHS starts showing around 350 particles, which in this case, it is behaving quite differently than the other two methods. The simulation results suggest that UHS with small number of particles has the same performance that both HS and FastSLAM have while using more particles. Fig.5.14 compares
the real travel time from the start point to the end of the road. In this simulation, different number of particles was used to observe how filters are behaving around the process time. The experiment was conducted for 11 different points, 20 times each. The traveling process time consumed by FastSLAM is the lowest, especially, with low number of particles. UHS on the other hand, consumes a longer time to process filtering and traveling even though the number of particles used by UHS is the same as the ones used by FastSLAM. HS process time is located between FastSLAM and UHS time, but closer to UHS; specifically in applications where a large number of particles are needed. The specific value of this time consuming process will depend on the type of application. Results show that even with the use of large amount of particles the difference in time would be between 1000 to 1500 seconds [78].

![Performance of Three Filters](image)

*Figure 5.13: Decreasing of RMS with increasing of particles [78]*
5.2 Loop Closing

In this chapter, many simulations are performed in order to address the problem of loop closing. As discussed in chapter 2, in the loop closing situation, with the assumption of correct data association, the challenge is to re-adjust the map optimally with all observation data gathered up to that point. This is specifically important in large maps where the filter is not able to remember uncertainty in the system and becomes overconfident. Structure of UHS is built to intentionally ignore all correlation tracking when the loop is closed. When the landmark at the closing gate is re-observed, its estimated location is compared to the very first coordination of the observed landmark at the beginning of the loop. There will normally be a
considerable error which is remembered through algorithm to be compared with the second set of error. If the difference is to some acceptable range (previously given to the algorithm), one of the estimated locations out of the two pieces of data is chosen randomly, otherwise, the first observation data will be incorporated in to the calculation. When the global map is constructed by the UKF, the difference between the observed location of a landmark and its predicted location must be as low as possible. This difference is usually referred to “innovation” or “residual” as described in section 3.1.3.3. The main use of the innovation is to correct the mutual relationships between every pair of landmarks. Unlike other discussed filters, UKF is able to correct the map configuration when the loop is closed and the first landmark is re-observed. Figure 5.15 demonstrates a scenario before and after closing a loop in which the robot completes the trajectory and let Unscented HybridSLAM estimate a map of the environment and location of the robot. The vehicle speed is maintained at 2m/s and there are 75 detectable landmarks in the environment. Same as the previous simulation scenario, number of particles is 200 in this particular case.

Figures 5.16 depicts absolute error and deviation along x and y axes. The dashed line represents the 1σ estimated uncertainty along x and y axes. The solid line represents the actual error along x and y axes. As it is expected, the largest error in trajectory occurs at the beginning of the process and in the centre of the map where the robot starts turning counter clock wise. However, the variation in the estimation error in both figures is increasing over time. The sub-filter “UKF” is aware that the presence of uncertainty would cause a substantial difference between predicated and true state estimation. UKF’s role is to estimate the variance in the state estimate through which the error in the state estimate elements is measured. As well, the absolute error and deviation is depicted in figure 5.17 for the orientation [79].
Figure 5.15: Loop closing scenario a) before closing the loop b) after closing the loop [79]
Figure 5.16: Absolute error and deviation along a) x axis b) y axis [79]
5.2.1 Consistency

The amount of innovation indicates how accurate the state estimate is. If the residual is equal to zero, it means that the measurement prediction is the same as the true state of the system. Should there be a difference between the predicted and observed measurements the state estimation has to be updated in the update step. The standard Kalman filter computes the variance in the state estimate, whereas the variance should at all time include the true state as the possible state. In order to evaluate UHS consistency, deviations of the state estimates are plotted as depicted in figures 5.18 and 5.19. Figure 5.18 illustrates the angle innovation within
the 2σ lines. The deviation shown in these figures are the square roots of variances, showing
the certainty that the true state lies within a certain distance from the estimated state.
In figure 5.19-a, UKF is 66% certain and is called the 1σ confidence interval. In figure 5.19-b,
the UKF is 95% certain and is called 2σ confidence interval, meaning that the true state
element lies within two deviations from the estimated element. 1σ and 2σ confidence
intervals plotted in these figures show that for the UKF as the sub-filter of UHS, the error lies
between the deviation lines. In the particular scenario depicted in figure 5.15, the error
remains well within the 1σ uncertainty lines, and even better between the 2σ lines all time
during the estimation process. This would imply that the filter is consistent, more specifically,
for the whole loop closing scenario duration [81].

Figure 5.18: Innovation and standard deviation for orientation in 2σ confidence interval [79]
Figure 5.19: Innovation and standard deviation a) $\sigma (65\%)$ b) $2\sigma (95\%)$ [79]
Figure 5.20 illustrates the standard deviation of a few landmarks for the same scenario in 10 seconds of the process. For the landmarks that are out of reach, their estimated errors are not continuously updated with the UHS. The filter updates a landmark as soon as the second pack of data is incorporated into the calculation. At this time the landmarks outside the vicinity of the vehicle are updated in that time step and their estimated error become exactly equal to the Full SLAM. Hence, all data related to the observation of landmark at a specific time step makes the computational cost proportional to the number of observed landmarks in the local area. One advantage of such proportion is that the uncertainty is remembered in the system where it is shared among landmarks. These results confirm the fact that the UHS performs similarly to the standard algorithm by which a large map can effectively be closed.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure5.20}
\caption{Landmark deviation for the loop closing scenario [79]}
\end{figure}
5.2.2 Loop Size

The loop closing problem does not just concern the re-observation process. While, usually a small loop would not create much of estimation error, the challenge is to keep the filter consistent when the loop becomes larger. This implies the use of greater number of particles for the calculation. However, the accuracy in estimation is not just dependent on the number of particles used in the calculation as there might be many other factors involved. These factors could be parameters of motion and observation models, number of landmarks in the environment, the level of ambiguity of observing landmarks, and even how distant those landmarks are. One of the main goals of a desired algorithm is to optimize the estimation process as much as possible [79].

Figure 5.21 compares the UHS accuracy in closing a loop with HS and FastSLAM. As shown, the RMS position error is increasing as the loop becomes larger. There is a threshold for each filter indicating how much the loop can be enlarged. For the simulation in figure 5.21, 100 particles are used as the base and only 100 landmarks are considered in the environment with 100m$^2$ area. For every 100m$^2$ increase of area, 100 landmarks are added to random points in the environment. The reason for such increase is to maintain the environment with the same distribution of landmarks. Results show that while for such increase in the area and number of landmarks, the accuracy of calculation decreases, the consistency of UHS is maintained up to 1000m$^2$ of area. HS becomes inconsistent around 500m$^2$ of area, and this number for FastSLAM is 200m$^2$. UHS appears to keep its consistency for a larger area compared to the other two filters. This result is due to less data ambiguity in UHS as the filter structure allows less ambiguity when it confronts a relatively larger loop [79].
5.3 Evolution of the Map

The range bearing sensor returns the distance between the sensor and the landmark as well as the angle between the robot’s frame of reference and the landmark’s frame of reference. Since all landmarks are considered static in the environment, the reference frame of landmarks is the same as global frame of reference. The measurement information related to a landmark is in form of polar measurement \((D, \beta)\), indicating range and bearing angle of the landmark. In the global frame of reference, the landmark coordinate is referred by \((x, y)\). In a standard filter (such as EKF) the first order Taylor series truncation is used to linearize this nonlinear...
observation model by which the calculation of any possible error becomes significantly inaccurate.

### 5.3.1 Robustness

In figure 5.22, the landmark position located at \((x=50.2, y=149.8)\) is estimated by incorporating range and bearing data \((D, \beta)\) received by the range/bearing sensor. In this case, zero-mean Gaussian noise is added and the data is converted to the Cartesian plane. As a result of low accuracy in estimation of the bearing direction, samples turn out to be in a banana-shape. As shown in figures 5.22-a, 5.22-b, and 5.22-c, the estimated mean and covariance of this distribution is relatively far from the true mean and covariance of the system. The estimation of the mean and covariance using Sterling approximation in the UHS algorithm is shown in figure 5.22-d which is the most accurate estimation among all filters. The inaccuracy in estimation of the mean in the range and the covariance in FastSLAM algorithm, usually results in the over estimation of the posterior distribution in the range which leads the filter to become over confident. HybridSLAM would produce an inaccurate mean and covariance since the linearization is based on the first order of Taylor series. As long as the map has a small size, all filters tend to approximate the mean and covariance of the system reasonably. Once the track becomes large, EKF fails to estimate an accurate map and both FastSLAM and HS become overconfident. Unlike the other three filters, UHS carries minimum error in the mean and the covariance estimate, leading to more consistency of the filter. The key to success for UHS is the Sterling approximation approach which results in high accuracy and moreover the robustness of the filter.
Figure 5.22: Estimation of the mean and covariance of a landmark a) EKF-SLAM b) FastSLAM c) HybridSLAM d) Unscented HybridSLAM
5.3.2 Map Fusion

In order to demonstrate how the map is evolved in the estimation process, a scenario is simulated in this section. Figure 5.23 illustrates an environment with a few landmarks gathered in small groups. The velocity is set on 2 m/s and the range for the range/bearing laser is 10 meters. As the vehicle starts localizing from point (0, 0), two landmarks at locations (x=2.3, y=5.2) and (x=5.0, y=3.6) are observed by the range finder. These landmarks are the first features observed in the vicinity of the vehicle, building the first piece of the map using sub-filter FastSLAM. At the same time, sub-filter UKF is building the same map independently. Once the first piece of the map is estimated by both suboptimal filters, the map built by FastSLAM will be added to the global map and at the same time, the uncertainty difference between sub-filter estimations will be taken into account for the CLSF. Once the robot observes first set of landmarks in the vicinity, the first region is added to the global map. The CLSF constraint would compare the map estimated by the particle filter to a map produced by UKF sub-filter. This is the very important stage to choose one map estimated over the other one. Depending on how accurate the map is constructed, a final decision is made and the first local map is added to the global map. In figure 5.24 the region added to the map is shown by a rectangle and the new region is under observation. Robot is approximately at the coordinate (10, 0). The first region inside the square is already a part of the global map. The robot is observing the second region consisting of only one landmark. Same as the first region, the map of the new region is built by both suboptimal filters and through CLSF the decision is made to add the most accurate map to the global map. Once the second region in the vicinity of the robot is estimated by the sub-filter FastSLAM, it is evaluated by CLSF which has already included the map by UKF. The decision is now based
on the CLSF algorithm evaluation to compare the two maps built by different suboptimal filters. If the map built by UKF is not crossing its threshold, it remains as is. If CLSF figures out that the map built by FastSLAM is more accurate relative to the one built by UKF, the accurate map is used. Figure 5.25 demonstrates the new area surrounded by a plaque and added to the global map. The first and second regions of the map are identified and part of the global map. The robot starts observing the third region from point (23, 3) with respect to the global map. The first and second regions are now correlated and have a critical role for the next decision making by CLSF. It should be noted that at this point, the first region is correlated with the second region and as a whole map, regions are expecting for the third region to join them in order to complete the map up to that time step.

Figure 5.23: Robot at coordinate (0,0).
Figures 5.26 to 5.28 demonstrate the remainder of the map evolution as the robot travels through the environment. Every time that a new region is estimated in the vicinity of the robot, a piece of the map is completed and added to the global map. It is CLSF task at every time step to determine the best evaluated sub-map fused into the global map. It is worth to note that as the vehicle travels along the path, the uncertainty in the position of the robot and location of landmarks increase as well. In figures 5.23 to 5.28, red ellipses indicate the uncertainty at each time step. The uncertainty increases over time, while the estimation in both locations of the robot and landmarks remain accurate.

![Map Evolution (UHS)-First region is covered](image)

*Figure 5.24: Robot is approximately at the coordinate (10, 0).*
In UHS algorithm, the local mapping is occurring and at the same time the map built by UKF is supporting the whole mapping process. Usually in the EKF-SLAM sub-algorithm of HS, spurious observations are associated with landmarks, particularly during transient periods of high vehicle uncertainty.

![Map Evolution (UHS)-Second region is covered](image)

*Figure 5.25: Robot starts observing the third region from point (23, 3) with respect to the global map.*

It should be noted that the Failures as the result of the EKF filter in HS does not happen in the proposed algorithm in similar scenarios. Even though the landmark is not observed simultaneously, the ambiguity can be resolved. The local linearization error is another failure
mode of EKF-SLAM which is not occurring when it is replaced by UKF. When the UKF algorithm begins to build a local map, the linearization error is at its minimum level. As a result, the uncertainty becomes small. In such case where the local-map uncertainty can be significant, the robustness of the UHS would cover any possible fault by integrating spurious observations of data association information of the observed landmark over time.

![Map Evolution (UHS)-Third region is covered](image)

*Figure 5.26: Robot is approximately at point (33, 26) and observing the fourth region in the vicinity.*

In figure 526 the correlation of three identified regions of the map is shown. The robot is approximately at point (33, 26) and observing the fourth region in the vicinity. Same process
will follow to add a new piece to the map. The third region is indicated by a trapezoid and contains two individual landmarks.

In figure 5.27 the robot is at the approximate coordinate of (25, 36) observing the fifth region of landmarks. The fourth region inside the circle is already part of the map and in correlation with the other three regions. There is a slight deviation between region 4 and region five which will be reduced once the robot is done observing region five. This will affect the RMS position error and orientation error and the goal for next step is to update the map and correct the path and heading estimation error.
Figure 5.28: The fifth region containing four individual landmarks is added to the map and correlated with other regions mutually.

In figure 5.28 the fifth region containing four individual landmarks is added to the map and correlated with other regions mutually. As the robot moves forward in the environment, more regions are observed and added to the global map using constrained local sub-map fusion algorithm as a sub-map strategy to build a unique map out of estimations made by two suboptimal filters. As indicated by arrows, the error becomes slightly shorter when the fifth region is added to the map. The RMS position error and the orientation error appears to increase before the new region is to be identified and added to the map.
5.3.3 Map Update

Figure 5.29 demonstrates a loop closing scenario in a domestic environment utilizing edges as line segments. The environment is a 50m by 50m area and the starting point is set on the global coordination (x=11, y=29). Edges are composed of approximately 1.00m distant landmarks by which lines at the corners of the environment are represented. In this simulation, map update process is demonstrated through sequential figures. The sensor which is installed in front of the robotic vehicle (shown in figure 4.2) returns the range and bearing information to landmarks in the environment. The sensor range for this simulation is set to 20 meters and vehicle speed is set to 2 m/s.

![True Map of the Environment and Waypoints](image)

*Figure 5.29: True map of a 50m by 50m environment in which waypoints are set. Waypoints are connected by straight lines showing an imaginary path.*
As shown in figure 530, the robot arrives at waypoint 2, estimating and updating the map of the environment. Black dots around landmarks indicate estimated locations of landmarks that are not incorporated into the map fusion process. At some point, the location of a particular landmark is over-estimated either by the FastSLAM sub-filter or by the UKF sub-filter. In this case, CLSF decides not to fuse that information into the global map and instead, updates the map with the previous data. On the other hand, if the estimation is not beyond some threshold, location of a landmark is fused to the global map, and as a result, becomes part of the absolute map that the system can rely on for the next time steps.

Figure 5.30: Robot is at the second waypoint
In figure 5.31, the robot is arrived at the sixth waypoint. More landmarks at the range of the sensor are observed and the map is still built using both suboptimal filters and with a final supervision of CLSF. As the process goes forward, landmarks that have been observed in previous time steps are re-observed from the new robot locations, and help the system ignore previous wrong estimations. Black “dots” and red “crosses” in this picture are changing their locations as the process proceeds. Figures 5.32 and 5.33 illustrate the resulting map at the tenth waypoint and at the end of the loop closing. Figure 5.34 demonstrates the final version of the map built during the loop closing indicating that the map update process has been successful using UHS. This map is now considered as the basis for global map referenced for the next cycle.

Figure 5.31: Sixth waypoint is reached by the robot and the map is built by suboptimal filters and the CLSF.
Figure 5.32: Robot is at the tenth waypoint

Figure 5.33: End of the first loop
Figures 5.35 to 5.36 show the RMS position and orientation error regarding the scenario depicted in figure 5.33. For this simulation, 300 particles are used and the vehicle velocity is set to 1.5m/s. RMS position error remains around 0.35 in average and the orientation error, in average is around 0.02 radians. The UHS algorithm appears to perform with a high accuracy in this case. Once the loop is closed, the map produced at the end of the loop will be used as a referenced map (global) for the next loop. It is very important to mention that at the beginning of the second loop, the SLAM problem may be treated as just a localization problem. In a standard localization problem, a map and the initial location of the robot are known and the goal is to keep track of the position while following a priori map. For the second loop closing, the robot does not usually need to imply SLAM algorithm, and it is possible to apply a simple
localization algorithm to use the previously produced map as a priori map for the second loop closing.

Figure 5.35: Orientation error is 0.02 radians in average for the SLAM process using UHS

Figure 5.36: RMS position error is 0.35m in average for the SLAM process using UHS
5.4 Distribution of Landmarks

The main task of a feature-based SLAM algorithm is to estimate the robot path and the environment map as accurate as possible. There are many methods in which the robot uses different sensors to measure landmarks’ positions as well as pose of the robot. In order to determine the current position of the robot, sensor readings are analyzed in these methods to extract data from the active or passive features in the environment to match it with a priori known information. Usually, the task of extracting and matching data with a priori information is straightforward for a domestic environment in which landmarks are distributed evenly.

If the robot has a notation of uniformly distributed landmarks in an environment, the extracting of such data would be rather easier than an environment with randomly distributed landmarks. In the next sections, an autonomous mobile robot is traveling in an environment in which landmarks are distributed uniformly and then randomly. The performance of the filter is compared in terms of data ambiguity in both cases.

5.4.1 Uniform Distribution of Landmarks

For some SLAM cases in which the robot is equipped with restricted sensors, a uniform distribution of landmarks would considerably reduce the data ambiguity associated with landmarks in the environment. The most interesting advantage in such cases would be the elimination of the data extracted from wrongly observed landmarks. As a consequence of the Maximum Likelihood Rule, since the robot is aware of a uniform set of landmarks, sensor readings that by pass a specific threshold would be automatically deleted from the estimation process.
Figure 5.37: Uniform distribution of landmarks in the environment

Figure 5.37 demonstrates one instance where a robot is traveling among uniformly distributed landmarks. Red dots around landmarks in this figure illustrate estimated positions of landmarks. In figure 5.38, a detailed estimation of the landmark located at \(x=30\)m and \(y=20\)m is demonstrated. 200 particles are used in this experiment. There are 64 detectable landmarks in the environment, and up to the observation time of the landmark \((x = 30, y = 20)\), 35 other landmarks in the neighborhood are observed. The estimated data beyond the threshold of observing the landmark \((x = 30, y = 20)\) is \(\pm 0.30\)m which is trivial and shows that the filter is properly capable of performing the estimation task. If by any chance the sensor mistakenly observes landmarks, the preliminary information of uniformly distributed landmarks will correct the misreading, resulting in the elimination of the wrong data. A notation of uniformly...
distributed landmarks is particularly important for applications in which the sensor is noisy. In that case, uniform distribution of landmarks notation would help the algorithm in order to perform the estimation task less erroneous.

![Figure 5.38](image.png)

*Figure 5.38: Estimated position of the landmark located at (x= 30, y =20) is shown. The error varies from -0.30m to + 0.30m along both x and y axes.*

### 5.4.2 Random Distribution of Landmarks

The task of algorithm to estimate position of a landmark becomes relatively hard if landmarks are not distributed uniformly. Due to the uncertainty in the pose of the robot and the high correlation between robot pose and location of the landmark, the error is unavoidable. If landmarks are distributed randomly, the data ambiguity can be induced easily and consequently, the estimation process becomes corrupted using different algorithms as explained in appendices A and B.
Unscented HybridSLAM algorithm is capable of handling the estimation process with an acceptable range of error in an environment with randomly distributed landmarks. Figure 5.39 illustrates the same path exemplified in section 5.4.1, but with a non-uniform distribution of landmarks. Figure 5.40 depicts a range of estimation for the same landmark located at (x=30,y=20). The error in this case indicates that the estimated location of the landmark is within ±0.40m along both axes which further demonstrates that the level of data ambiguity does not arise exponentially when landmarks distribution change from uniform to random.

*Figure 5.39: Non-uniform distribution of landmarks in the environment.*
Figure 5.40: Estimated position of the landmark located at \((x=30, y=20)\) in a non-uniform distribution

### 5.4.2.1 Threshold

Figure 5.41 compares the ambiguity of data applying EKF-SLAM and three other algorithms using 3000 particles; FastSLAM, HybridSLAM, and Unscented HybridSLAM. In this figure, Hundreds of dots that make different formations around the range are depicted for each specific filter. The threshold oval is obtained using a standard EKF under Gaussian conditions. The true landmark position is at \(x=30\text{m}\) and \(y=20\text{m}\). The banana shape in figure 5.41-a display the estimation result using the first order Taylor series in EKF under non-Gaussian conditions which appears to be highly inaccurate.
Figure 5.41: Estimated position of the landmarks compare to the standard threshold. a) EKF-SLAM under non-Gaussian conditions b) FastSLAM c) HybridSLAM d) Unscented HybridSLAM [77]

The banana shape in figure 5.41-b, illustrates an error reduction in the location estimation of the landmark using FastSLAM, resulting in less ambiguity in data [76]. However, estimated points do not fit in the standard oval and there are about 60% of estimated points off the standard threshold. As shown in figure 5.41-c, HS has a relatively less ambiguity in data association. Apparently, there are only 30% of points off the range. Moreover, even though the orientation of estimated points is more towards the orientation of the standard oval, it is
still far from the standard threshold and may not be an acceptable result for SLAM applications. The estimation of the landmark using UKF that forms an oval around the true location of the landmark is the one with the least ambiguity in data association. As shown in figure 5.41-d, about 15% of estimated points is off the standard threshold which allows for it to have the most acceptable result amongst all filters. As a result, UHS is the only recursive filter based on sterling approximation and has the least tendency to diverge. Figures 5.42 to 5.45 demonstrate the performance of UHS for the scenario depicted in figure 5.39 [77].

![Figure 5.42: Comparing the absolute error and deviation in terms of orientation [77]](image-url)
Figure 5.43: Absolute error and deviation along a) x axis b) y axis [77]
Figure 5.44: Deviation of landmarks \((x=10, y=0)\) and \((x=20, y=0)\) using 3000 particles [77]

Figure 5.45: Deviation of landmarks \((x=30, y=40)\) and \((x=12, y=48)\) using 3000 particles [77]
5.4.2.2 Double Loop

This section presents simulation results of a double loop scenario using Unscented HybridSLAM algorithm. The double loop case is considered in order to analyze the performance of the algorithm when it is facing a more complicated terrain [77]. Figure 5.46 shows the environment map which contains a non-uniform distribution of point feature landmarks as well as the true path of the robot. The speed of the Mobile robot is considered 3.5 m/s and the process time to complete the whole loop is approximately 2800 seconds, while number of particles used in this experiment is 500 [77].

Figure 5.46: True map of the environment with 94 observable landmarks [77]
Figure 5.47 depicts the true map of the environment and observation results prior to closing the whole loop. The vehicle starts at the centre of the environment \((x=0, y=0)\) and travels counter clockwise. As the robot travels around the path, more landmarks are observed and the uncertainty increases slightly. The uncertainty in the observations is at the largest value on the third part of the main loop. Figure 5.48 demonstrates the actual error and standard process deviations up to the third quarter of the loop. Blue and red solid lines in this figure illustrate the actual error in \(x\) and \(y\) positions respectively. Dashed lines represent the \(1\)-sigma estimated uncertainty. The simulated result indicates that UHS is consistent with the actual error up to the point \((x=12, y=18)\) [77].

![Graph showing robot's travel and observation results](image)

*Figure 5.47: Robot is traveling CCW and has observed landmarks up to three quarter of its total map [77]*
Figure 5.48: Absolute error and deviations along x and y axes [77]

Figure 5.49 depicts the evolution of the uncertainty for 4 out of 6 landmarks inside the internal loop at the beginning of the process. All solid lines represent the deviations and dashed lines represent the error. Comparing the error between the actual landmarks positions and those estimated with the 2-sigma deviations indicate that the UHS filter is consistent when facing a double loop case, specifically with respect to landmarks error. As expected, the actual landmarks error and uncertainty are reduced. Two out of six landmarks in the internal loop were not observed due to the range limitations. Figure 5.50 shows the result with respect to the orientation deviation and absolute error right after the loop is closed and indicates that the map becomes more correlated at the end of the first run [77].
Figure 5.49: Landmark deviation and absolute error (double loop case) using 500 particles [77]

Figure 5.50: Orientation Absolute error and deviation (double loop) using 500 particles [77]
Figure 5.51 depicts the situation in which the loop is already closed and the robot has traveled one third of the trajectory. In this figure, the mobile robot is at point \((x=-20, y=34)\), heading to finish the second loop. The uncertainty in the landmarks observation at this point is substantially reduced, meaning that the loop closing outcome was successful and the filter has converged. Moreover, all observable landmarks are estimated correctly following the completion of the first run. Figure 5.52 is a close up of the uncertainty on the path of the vehicle and demonstrates that UHS is generally tractable for a large and double loop scenario. In this figure, the RMS position error is around 0.3m which indicates the map has become correlated along the path [77].

![Figure 5.51: Robot is still traveling CCW after the completion of the loop located at point \((x=20, y=34)\) [77]](image-url)
In figure 5.52 the robot is at point (x=20, y=34) and has almost completed one third of the loop after closing the loop. The red line indicates the estimated path, blue line is the true path, and the red border oval represents the uncertainty in the path estimation. The estimated uncertainty is consistent with the actual error for x and y axes as well as the heading angle of the vehicle [78]. Figures 5.53 and 5.54 show the absolute error and deviations along both axes x and y, and for the orientation. Figure 5.55 shows the absolute error and deviations for six landmarks inside the internal loop, right after the loop completion when the robot is at the third of its path during completion of the second loop. The evolution of the uncertainty for all six landmarks in the map indicates that the map correlation is maintained and proves that the final map is consistent. Results indicate that the estimated uncertainty is consistent with the
actual error along both axes as well as the orientation of the vehicle. The orientation error shows an average error of 0.02 radians which guarantees the consistency.

Figure 5.53: The loop is completed and the vehicle is at one third of the path on the second run [77]

Figure 5.54: Orientation Absolute error and deviation (double loop case) using 500 particles [77]
Figures 5.56 and 5.57 depict the 2-sigma standard deviation (95%) vs. the innovation in terms of vehicle heading angle as well as its distance. Simulations show that the mean remains close to zero which results in a reliable and consistent map even though the loop is completed and the vehicle is on its second run of loop completion. Comparing the outcomes from a double loop case with a regular loop proves that as long as the ambiguity of data association is reduced by the UHS, a double loop does not affect the performance of the filter, and the results would be identical to a SLAM case with a regular loop closing case. This is important since filters such as EKF fail to fulfill the navigation task in such circumstances [79].
Figure 5.56: Standard deviation vs. distance innovation [79]

Figure 5.57: Standard deviation vs. angle innovation [79]
5.5 Future Work

This dissertation may be used as a theoretical basis for further research in the field of Simultaneous Localization and Mapping problem. During this research, many ideas sparked for the improvement in the SLAM field. Most SLAM methods implement the Bayesian representation. While a Bayesian approach is currently the most suitable representation to acquire alternative solutions, different approaches may be subject of future research when envisage different applications and different environments.

Further studies are recommended with regard to landmarks representation in a feature-based SLAM problem. Even though, there are different techniques other than the ones in this thesis resulting in more efficient solutions with less ambiguity in data association, they have proven to be less cost efficient and might not be suitable for real time applications. Future study on combining such methods in order to make them applicable for real world scenarios is also recommended. Also, with the basic derivations of Unscented HybridSLAM, it is interesting to look for more details into other extensions.

World is not static and SLAM problem need to be solved in dynamic environments. In this thesis, solutions to SLAM problem were discussed with non-moving features. It is of the most interest in SLAM applications to look at a scenario in which pose of a landmark is a function whose variable is time. Of such interests, people and moving objects tracking, interactive navigation, and multiple robot SLAM may be the ultimate goal of autonomous mobile robots navigation. Future studies may focus on designing a high level algorithm which can handle complicated models of motion and observation to obtain the most optimum solution with more efficient costs.
Finally, the validation of the proposed filter Unscented HybridSLAM was done in this study using a simulator, and merely meant as illustrations to concepts of filtering and localization and mapping estimation. In order to draw practical conclusions, the theory discussed in this thesis should be implemented with a real platform to investigate its validity, reliability, and applicability along with proper investigation on the necessary modifications and improvements.
Chapter 6

Conclusions

This chapter concludes the study presented in this dissertation and draws main contributions made in this research. In this dissertation a new approach to the Simultaneous Localization and Mapping problem called Unscented HybridSLAM is presented. The new approach combines two strategies and anchors their advantages to improve the correct correspondence between path of the robot and the constructed map during the estimation process. The main contribution in this study is the combination of two currently used strategies, avoiding their shortcomings, and improving the computational efficiency of the estimation problem involved in either strategy; FastSLAM and Unscented Kalman Filter. The proposed algorithm is modified theoretically to identify correlations between estimate errors in motion and observation. Afterwards, the convergence of the filter is mathematically proven and guaranteed not to diverge.

One of the most important contributions of this thesis is the use of Constrained Local Sub-map Filter as a strategy to fuse the local map estimated by FastSLAM into the global map estimated by Unscented Kalman Filter. The lower bound goal on map accuracy using Constrained Local Sub-map Filter strategy is thoroughly investigated to construct a reliable map based on map fusion techniques. With the abilities of the Constrained Local Sub-map Filter techniques, the proposed algorithm takes advantage of both suboptimal filters FastSLAM and Unscented Kalman filter strengths to reformulate the SLAM problem and to reduce the data association ambiguity problem and thus, provides a computational tractable solution to many loop closing scenarios. The construction of feature-based SLAM using Rao-Blackwellised particle filter, Unscented Kalman filter, and Constrained Local Sub-map
Fusion, results in a novel approach (Unscented HybridSLAM) which is able to handle potentially a large number of observations with a low bound ambiguity.

Another major contribution of this thesis is the development of a theoretical Bayesian representation of the SLAM problem minimizing the computational complexity of the probabilistic estimation based approach, leading the problem to acquire a tractable solution in different scenarios, specifically, when the motion is highly non-linear. Probability density functions of the motion model and the observation model are developed based on a Bayesian representation to transform the continuous motion as a parameterized probability form, making computations tractable during the estimation process. A car-like mobile robot close to a realistic outdoor platform is exemplified and the equations of motion is derived and transformed to a linear form for simulation experiments. Subsequently, Jacobian matrices are used to linearize both motion and sensor models. Subsequent to the linearization process for the first term of Taylor series, the second order Sterling Polynomial Interpolation is used to linearize a highly non-linear motion through the second term of Taylor series.

The final contribution of this thesis is a thorough investigation and the validity check of the proposed algorithm with simulation results. Using MATLAB codes, simulations of the vehicle trajectory and the estimation accuracy are investigated based on proposed algorithm. The map accuracy and path estimation are compared to currently used algorithm, in particular, resulting in high quality of the produced map and reducing error in estimated position and robot heading. Results from different scenarios indicate that the proposed algorithm is be able to handle hundreds of nearby landmarks with minimum data association ambiguity and a high level of accuracy in the path estimation when compared to currently used algorithms. Different simulations performed, demonstrate that Unscented HybridSLAM outperforms its
suboptimal filters by employing their advantages in the estimation process while avoiding their shortcomings. The robustness of the filter is experimentally validated through which makes Unscented HybridSLAM algorithm a role player for future real world applications.

The material covered in this dissertation forms a basis for future studies in the field of Simultaneous Localization and Mapping Problem and autonomous mobile robots navigation. While the practical aspects of Unscented HybridSLAM may be applicable in different domains, the subject of this study focuses in the field of autonomous robots and may, in particular, set the stage for future studies in a wider range of autonomous navigation systems.
References


[51] J. Guivant, E. Nebot, “Compressed filter for real time implementation of simultaneous localization and map building”, Australian Centre for Field Robotics, Department of Mechanical and Mechatronic Engineering, University of Sydney, Sydney, Australia, NSW 2006.


Appendices

Appendices A and B focus on more details regarding the structure of suboptimal filters are used to construct and reformulate SLAM problem. Theses appendices give a thorough introduction of each algorithm structure which enables the reader to acquire a good basis in order to use when applying different solutions in different application domains. The computational origins will be discussed in great detail while paying attention to the general ideas and intuition behind different equations.

Estimation using UHS is adopted in this thesis as a reliable solution to SLAM problem. The formulated algorithm efficiently scales to cluttered environments with a high level of robustness when facing ambiguous data association. In order to develop Unscented HybridSLAM, a detailed description of its components is necessary. Therefore, the background knowledge of Unscented Kalman filter and Rao-Blackwellisation for a common Monte Carlo method will be necessary to understand how the combined filter is performing the estimation task. Ultimately, in the third chapter, combining Unscented Kalman filter (UKF) and FastSLAM using a Constraint Local Sub-map Fusion (CLSF) strategy, is demonstrated supported by a convergence proof for the FastSLAM as described in [8].
Appendix A

Suboptimal Filters UKF and EKF

A.1 Unscented Kalman Filter Algorithm

EKF is vastly used for a non-linear dynamic system in order to propagate the Gaussian random variable (GRV) through the system dynamics. The propagation is analytically through the first order linearization of the non-linear system to the first order of Taylor series expansion. As a result, a large amount of error is generated in the mean, and covariance of a posterior belief of the robot leading the system to a large amount of uncertainty at the time step k. In cases in which there is a high level of non-linearity in the system, the uncertainty will be substantially large at each time step. UKF is an improved version of linear KF proposed by Julier and Uhlman [46] and further developed by Wan and van der Merwe [48] which addresses this problem using a deterministic sampling approach. Same as EKF, a Gaussian random variable approximates the distribution of the system in UKF algorithm with the exception that a minimal set of chosen sample points are used to capture the true mean and covariance of the GRV to the second order of any non-linearity of Taylor series expansion accurately. In upcoming sections, the structure of the algorithm will be shown.

A.1.1 Noise Characteristics in UKF

There is always some noise around the sensing devices that are corrupting the navigation process. Same as the other extensions of Kalman filter, in a navigation process, the corrupting
noise must have specific characteristics. The corrupting noise has to be independent, white, and, zero mean, while this noise needs to be Gaussian to make the best estimator out of a Kalman based algorithm [26].

A.1.1.1 Independent Noise

The independency assumption makes the computation in a SLAM filter much easier. Generally, it is fair to assume that the observation noise and the motion noise are mutually independent; meaning that the amount of noise at the observation model does not have any influence on the motion model and vice versa. [59].

A.1.1.2 White Noise

The white noise assumption in the algorithm makes the computation relatively easy. White noise is a type of noise that has same power at all frequencies in the spectra and is completely uncorrelated with itself at any time but the current time step \( k \) [6].

A.1.1.3 Zero Mean Value

Zero mean assumption implies that the amounts of error in motion and measurement models are random. Random noise can be classified into systematic and non-systematic. Systematic noise is a type of noise that constantly corrupts the system state or measurements with a certain behavior. On the contrary, a non-systematic noise is not predictable and varies in a large interval without having any domestic function to mathematically describe it. A random
error whether systematic or non-systematic, may at times be positive or negative. The generated noise in most today equipments can be expressed as a systematic noise [50], as assumed in this thesis.

A.1.2 Gaussian Modeling

The state of a dynamic system consists of a vector of position and orientation of the robot and positions of landmarks. The noise which is corrupting the system generates erroneous results in relative and absolute measurements and if ignored, the system obtains a significant amount of uncertainty [26]. A Kalman filter-based algorithm converges only under Gaussian implications meaning that motion, Observation, and noise have to be all expressed as Gaussian distribution functions.

A.1.2.1 Noise Gaussian Representation

The Gaussian assumption of noise makes the computation of a filter tractable. This assumption states that the amount of noise can be modeled as a bell-shaped curve and is justified by supposing that motion and observation noise often originate from multiple small noise sources. Regardless of how the source of noise may be distributed, the sum of all these independent sources can be considered as a Gaussian distribution function [60]. Moreover, the computation of the filter with the Gaussian assumption becomes easy and the reason is that only the first and second order statistics of noise characteristics (i.e. mean and variance) are known. Many measurement devices provide only a nominal value of the measurement.
A.2.1.2 Motion as a Normal Distribution Function

To represent a random noise in a Gaussian form, some assumptions should be considered prior to using UKF algorithm. With the zero-mean and Gaussian distribution assumptions for the motion noise, the distribution of corrupting noise may be expressed as

\[ w_k \sim N(0, Q_k) \]  \hspace{1cm} (A.1)

where \( N(0, Q_k) \) denotes the Gaussian function with zero-mean and the system noise covariance matrix \( Q_k \) at time step \( k \) [6]. The system noise covariance matrix \( Q_k \) can be described as

\[ Q_k = E[ (w_k)(w_k)^T] \]  \hspace{1cm} (A.2)

It should be noted that the main diagonal of the covariance matrix \( Q_k \) contains the variance in the state of the system.

A.2.1.3 Observation as a Normal Distribution Function

Similar to motion model, observation sensors are always subject to noise. This noise is considered random, independent, white, and with zero-mean [6]. Similar to the motion model, the observation noise may be expressed as

\[ v_k \sim N(0, R_k) \]  \hspace{1cm} (A.3)
where $N(0, R_k)$ describes the Gaussian function. This function has zero-mean with the system noise covariance matrix $R_k$ at time step $k$, where, $R_k$ can be described as

$$R_k = E[(v_k)(v_k)^T]$$  \hspace{1cm} (A.4)

The main diagonal of the covariance matrix $R_k$ contains the variance in the observation vector variables $v_k$.

### A.1.3 State Estimation

The approximation using UKF is more accurate in comparison with EKF. EKF employs first order of Taylor series expansion, specifically when motion and observation models are highly non-linear. In UKF, sigma points (minimal set of sample points) around the mean are used in a deterministic sampling technique called unscented Transform [45]. By propagating sigma points through non-linear functions of motion and observation models, a high estimation of true mean and covariance of the system can be obtained [48]. It is important to mention that unscented transform eliminates calculation of Jacobians in UKF by which the complexity of computation is reduced substantially. However, the estimation is done in a prediction and update manner similar to EKF. Most dynamic systems are non-linear and since UKF needs non-linear descriptions of motion and observation models, they need to be expressed in terms of non-linear functions.
A.1.3.1 Nonlinearity of Motion

The evolution of the motion can be described in form of a probabilistic model. An adequate non-linear model to describe a real noisy dynamic system [6] is defined as

\[ P (x_k | x_{k-1}, u_k) \leftrightarrow x_k = f (x_{k-1}, u_k, w_k) \]  (A.5)

where \( f(.) \) is a non-linear system function that relates the state of the system in the previous time step \( k-1 \) to the current time step \( k \). Matrix \( u_k \) is the vector of control actions. \( w_k \) is the system noise with its covariance matrix \( Q_k \), white, and with a zero-mean Gaussian distribution function.

A.1.3.2 Nonlinearity of Observation

The sensor model describes how the observations are related to the states of the robot. A Kalman filter algorithm needs a sensor model to update the state estimation using the observation readings. Given the true state of the system, the observation can be described by a model that compares the real observation to the observation given by the model to ultimately correct the state estimation. Due to non-linearity of real systems, the probability density of the observation model is denoted in the general form of a normal distribution function as

\[ P (z_k | x_k) \leftrightarrow z_k = h (x_k, v_k) \]  (A.6)
where \( h(.) \) is a non-linear observation function which relates the state of the system \( x_k \) to the measurement. It is beneficial to remind that the observation noise \( (v_k) \) is independent from motion noise \( (w_k) \). Similar to the motion noise, the observation noise is considered Gaussian, white, independent and with zero mean. Matrix \( R_k \) described in equation (A.4) is the covariance matrix of the observation noise [6].

### A.1.3.3 EKF Algorithm

The block diagram in figure A.1 represents the relationship between control, noise, observation and motion [48]. In this diagram, the output \( (z_k) \) is the observation at time step \( k \). \( u_k \) inputs are the action that brings the robot from previous time step to the current and establishes the current state of the robot. Motion noise and observation noise are added to the system during the process. As discussed before, noise is white, zero mean, and with a normal distribution function. With the definition of the standard EKF the estimation of the state and its main covariance matrix of the posterior belief of \( Bel^+(x_k) = P(x_k \mid Z_k, U_k, x_0) \) can be computed as

\[
\hat{x}_k^+ = E [ x_k \mid Z_k ] \quad (A.7)
\]

\[
P_k^+ = \begin{bmatrix}
P_{x_k x_k}^+ & P_{x_k m_k}^+ \\
P_{x_k m_k}^T & P_{m_k m_k}^+
\end{bmatrix} = E[(x_k - \hat{x}_k^+) (x_k - \hat{x}_k^+)^T \mid Z_k] \quad (A.8)
\]
To estimate the state of a non-linear system with non-linear observations subject to Gaussian noise, EKF provides a recursive estimate of both state and covariance matrix. There are three steps for EKF to fulfill the task; prediction, observation and update.

**Prediction**-At time step $k$, the EKF propagates the state and uncertainty of the system at the previous time step. At this step the algorithm first generates a prediction for the state estimate using equation (A.9), following the state prediction, the predicted observation relative to the $i^{th}$ landmark is done using equation (A.10) and then the state estimate covariance prediction is computed using equation (A.11).

**System Model**

$$\dot{x}_k^- = \Phi_k \hat{x}_{k-1}^+ + u_k$$  \hspace{1cm} (A.9)

**Sensor Model**

$$\hat{z}_k^- = h(\hat{x}_k^-)$$  \hspace{1cm} (A.10)

$$P_k^- = \Phi_k P_{k-1}^{+} \Phi_k^T + Q_{k-1}$$  \hspace{1cm} (A.11)
The Jacobian matrix $\Phi_k$ contains the partial derivatives of system function $f(.)$ with respect to state $x$ evaluated at the posterior state estimate $\hat{x}_{k-1}$ in time step $k-1$.

$$\Phi_k = \left. \frac{\partial f(x)}{\partial x} \right|_{x = \hat{x}_{k-1}}$$  \hspace{1cm} (A.12)

**Observation**- Following the prediction step, an observation $z_k$ of $i^{th}$ landmark is made according to equation (A.6). The difference between the observation $z_k$ and the prediction observation $\hat{z}_k$ is called the innovation or the residual $\tilde{z}_k$. The innovation denotes the difference between the predicted observation and the real observation at time step $k$.

$$\tilde{z}_k = z_k - \hat{z}_k = z_k - h(\hat{x}_k)$$ \hspace{1cm} (A.13)

If the innovation is negligible (or close to zero), the predicted measurement is considered the same as real measurement in equation (A.6). This means that the measurement of the state estimate at time step $k$, was very close to the true state in time step $k$. The innovation covariance matrix is then described as

$$\tilde{Z}_k = \Psi_k \ P_k \ \Psi_k^T + R_k$$ \hspace{1cm} (A.14)

$\Psi_k$ is the Jacobian matrix with partial derivatives of the observation function $h(.)$ with respect to the state $x_k$ evaluated at the prior state estimate $\hat{x}_k$. 

Update - This step is called the “observation update step” or “correction step” or simply “update step”. The following equations correct the most recent belief of the system $Bel^+(x_k)$ [61]. The step which is the final state estimate needed by the algorithm for completion, is critical. The equations are

$$\hat{x}_k = \hat{x}_k + K_k \tilde{z}_k$$  \hspace{1cm} (A.16)

$$P_k^+ = P_k^- - K_k (\Psi_k^T P_k^- \Psi_k^T + R_k) K_k^T$$  \hspace{1cm} (A.17)

The factor $K_k$ is called Kalman Gain (KG) [22] which determines the extent that the innovation should be taken into account in the system posterior state estimate. Kalman gain determines this extent by looking at the relative uncertainty between the prior state estimate and the observation innovation [45]. To compare the prior state estimate uncertainty in the state space with the innovation uncertainty in the observation space, Kalman Gain converts the uncertainty in the observation space to the state space by means of the matrix $\Psi^T$. Kalman Gain is expressed as

$$K_k = P_k^- \Psi_k^T (\Psi_k^T P_k^- \Psi_k^T + R_k)^{-1}$$  \hspace{1cm} (A.18)

$$\Psi_k = \frac{\partial h(x)}{\partial x} \bigg|_{x=x_k}$$  \hspace{1cm} (A.15)
Appendix B

Suboptimal Filter FastSLAM

B.1 Rao-Blackwellised Particle Filtering

To overcome shortcomings of EKF-SLAM, FastSLAM as an alternative solution to SLAM problem was introduced by Montemerlo [8]. FastSLAM made a revolutionary improvement in the design of recursive probabilistic SLAM and use of the filter with less concerns regarding EKF issues. FastSLAM, based on Recursive Monte Carlo Sampling, directly represents a non-linear process model which is not necessarily under Gaussian conditions. This method represents distributions using a finite set of “sample states” or “particles”. FastSLAM algorithm was built based on the earlier work of Thrun and colleagues [16, 30]. FastSLAM, as Monte Carlo Localization utilizes a set of particles to represent the distribution of possible states of a robot relative to a fixed map. However, this approach can be reduced to the sample-space by applying Rao-Blackwellisation (RB). Whereby a joint state is partitioned according to the product rule \( P(x_1, x_2) = P(x_2 | x_1)P(x_1) \). If \( P(x_2 | x_1) \) can be represented analytically, only \( P(x_1) \) needs to be sampled.

B.1.1 Data Association in Observation of Landmarks

In FastSLAM algorithm, the posterior is estimated over the landmark location, however, instead of the single state of the robot \( x^R_k \), the path \( X^R_k \) is considered. In SLAM, every individual landmark has some specific information that is obtained from the range\bearing
sensor observation and is unique, meaning that it is dedicated only to that individual landmark bearing and range [64]. This unique information of a landmark can be independently incorporated into the filter calculation. Therefore, the assumption is that each observation gives the location data for only one individual landmark \( m_i \) relative to the current state of robot \( x^R_k \) at time step \( k \). In a set of observation \( z_k = \{z_{k,i}, z_{k,j}, \ldots, z_{k,n}\} \) for different landmarks at time step \( k \), every member of this set is the observation of one individual landmark at that time step \( k \). For instance, if at time step \( k \) the robot observes landmark \( i \), the observation regarding that landmark will be indicated as \( z_{k,i} \). The information or data related to that individual landmark observation which is incorporated to the map calculation is identified by character \( d_{k,i} \). Here, a set of information or data obtained from observation of many landmarks at time step \( k \) is considered. This set specifies what information about what landmark and in what time step is being incorporated to the filter calculation. The set of all data associated with the map according to the set of observations can be presented as

\[
d_k = \{d_{k,i}, d_{k,j}, \ldots, d_{k,n}\}
\]

\( \text{(B.1)} \)

**B.1.2 Independent Landmarks in a Bayesian Network**

In figure B.1, a “Dynamic Bayesian Network” (DBN) is illustrated. \( m_j \) is observed at the first time step, \( m_i \) is observed at the first time step, time step 2 and time step \( k \) and finally, landmark \( m_n \) is observed at time steps; 2 and \( k \). This figure indicates that if the true path of the robot is known, landmarks \( m_i, m_j, \) and \( m_n \), are mutually independent. In other words, there will be no correlation between any two landmarks (two nodes \( m_i \) and \( m_j \)) of the map. This
property is called “d-separation” [63] in a DBN which makes the SLAM problem low-dimensional due to the fact that an observation of a landmark will not provide any information about the position of any other landmark and makes landmarks mutually independent.

![Dynamic Bayesian Network]

Figure B.2: A dynamic Bayesian network.

This leads the SLAM problem to the important fact that the information regarding the location of a landmark does not have any impact on the other landmarks information. Therefore, the path of the robot is known, and consequently, the location of the observed landmark can be estimated without the need of any information regarding any other landmarks. Since correlation between elements of the map only arises through robot pose uncertainty, if the true path is known, the landmark positions can be estimated independently. As a result, the SLAM posterior can be re-written as equation (1.1) previously discussed in chapter 1, which is a
factorization of particle filter for the pose estimation from a set of landmark estimation products. The factorization is described in next section and the derivation proof is presented in Chapter 3 based on work of [8].

**B.1.3 Factorization**

From Figure A.2, and using the definition of conditional probability, FastSLAM can be expressed as

\[
P(X_k^R, m | Z_k, U_k, x_0^R, d_k) = P(X_k^R | Z_k, U_k, x_0^R, d_k) P(m | X_k^R, Z_k, U_k, x_0^R, d_k)
\] (B.2)

Since one can write the second term of right hand side of equation (3.45) as

\[
P(m | X_k^R, Z_k, U_k, x_0^R, d_k) = \prod_{i=1}^{M} P(m_i | X_k^R, Z_k, U_k, x_0^R, d_k)
\] (B.3)

Therefore, the equation (3.45) can be rewritten as

\[
P(X_k^R, m | Z_k, U_k, x_0^R, d_k) =
\]

\[
P(X_k^R | Z_k, U_k, x_0^R, d_k) \times \prod_{i=1}^{M} P(m_i | X_k^R, Z_k, U_k, x_0^R, d_k)
\] (B.4)
This equation indicates that with the notification of robot’s path, a landmark position is conditional to path of the robot and independent of the other landmarks. As a result, there will be \( M+1 \) filters; one particle filter for path of the robot (path posterior) and \( M \) Extended Kalman Filters for landmarks positions estimation (landmark estimators) associated with the path \([8]\). “This factorization is absolutely exact and fits very well for any SLAM application” \([8]\). When the filter estimates path of the robot using particles, every landmark position is being tracked by EKF. Number of EKFs depends on the number of particles \( P \). As a result, there will be \( M \times P \) EKFs in total in order to estimate landmarks locations using path of the robot at each time step resulting the filter to have multiple hypothesis data association property.

### B.1.4 State Samples

The pose of the robot for a specific particle ‘\( n \)’ can be expressed as

\[
^nX_k = \{ ^nX_k^R, ^n\mu_{k,1}, \ldots, ^n\mu_{k,M}, ^n\sigma_{k,1}, \ldots, ^n\sigma_{k,M} \} \tag{B.5}
\]

where, \(^nX_k^R\) is the \( n^{th} \) particle’s estimation of the path. A sample of all particles at time step \( k \) is expressed as

\[
\text{Sample of } x_k^R = \{ x_k^R, 2x_k^R, \ldots, ^nX_k^R \} \tag{B.6}
\]
The superscript “n” on the left hand side of each term shows the involvement of the \( n \)th particle at time step \( k \) for estimating the path and \( M \) landmarks locations in the map using EKF, and according to the estimated path using RBPF.

### B.2 Data Association Assumptions

In this thesis, mapping between observation and landmarks is considered as known data. Known data association simply means that the observation information of each landmark is being incorporated to the map without being mystified with any other data observation of other landmarks. In other words, the data from each observation is considered known and related to only one specific landmark. The classic way of dealing with the data association problem in SLAM is to choose \( d_{k,i} \) such that it maximizes the likelihood of sensor measurement \( z_k \) given all available data.

\[
\tilde{d}_{k,i} = \arg\max P(z_k | d_{k,i}, \tilde{d}_{k-1}, X^k, Z_{k-1}, U_k)
\]  

The term \( P(z_k | d_{k,i}, \tilde{d}_{k-1}, X^k, Z_{k-1}, U_k) \) is called a likelihood, and in the event that the data association picks the maximum value, it is called the Maximum Likelihood Estimator (MLE). The assumption of known data association simplifies the computations which is straightforward. While this is not the case in most real world application, with the assumptions of known data association in FastSLAM, the cumbersome calculations can be avoided and the parameters of unknown data association to the computation can be added.
later [8]. Thrun and colleagues [36] have derived an improved algorithm regarding the
unknown data association for the current version of FastSLAM. Nonetheless, for simplicity
purposes, the data association is considered known for the calculation.

B.3 FastSLAM Algorithm

FastSLAM employs four steps to draw an estimated pose out of a series of samples. These
steps are “state sampling” (SS), “landmark estimation update” (LEU), “importance weight”
(IW), and “importance sampling update” (ISU) [8].

B.3.1 State Sampling

State sampling is the first step of FastSLAM algorithm. Before final prediction of the path,
pose or state of the robot must be estimated for the time step \( k \) and added to the path up to one
time step earlier (\( k-1 \)). This estimation is done for each particle belonging to the set of
particles, and is up to the time step \( k-1 \), where the pose has already been predicted.

Now a guess of pose must be done for the current time step \( k \). To do so, a probabilistic motion
model based on state of the robot at time step \( k \) is employed and expressed as

\[
^n x_k^R \sim P \left( x_k^R \mid u_k, ^n x_{k-1}^R \right) \tag{B.8}
\]

Estimation at this level is added to a temporary set of particles along with the estimated path
at time step \( k-1 \) which is \( ^n X_{k-1}^R \). Assuming that the sample of particles at time step \( k-1 \), is
distributed as \( P \left( X_{k-1}^R \mid Z_{k-1}, U_{k-1}, x_0^R \right) \), new particles distribution can be expressed as
Equations (B.51) and (B.52) are according to the distribution of particle filtering. The map size does not affect the particles sampling time consumption. Regardless of the map size, drawing the new pose of each particle is a constant time operation. Furthermore, the system does not have to be linearized. In fact, equation (B.50) can deal with any non-linearity of the system which can be a significant advantage of FastSLAM over EKF-SLAM. The distribution according to equation (B.52) is also referred to proposal distribution of particle filtering.

### B.3.2 Landmark Estimation Update

If the data association \( d_{k,i} \) at time step \( k \) regarding the observation of the \( i^{th} \) landmark \( m_i \) is known, the whole data association of observation of \( n \) landmarks at this time step \( k \) can be expressed as

\[
\mathbf{z}_k = \{ \mathbf{z}_{k,i}, \mathbf{z}_{k,j}, \ldots, \mathbf{z}_{k,n} \} \quad (B.10)
\]

where the observation of all landmarks up to time step \( k \) can be expressed as

\[
\mathbf{Z}_k = \{ \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k \} = \{ \mathbf{Z}_{k-1}, \mathbf{z}_k \} \quad (B.11)
\]

The known data association assumption is just for simplicity in order to derive equations of observation. Montemerlo has derived a modified version of Fast SLAM called FastSLAM 2.0
which deals with unknown data association as well. If there are $M$ observable landmarks in the environment, then $M$ low dimensional EKFs are considered for each of $P$ particles. In total, there will be $M \times P$ Extended Kalman Filters involved in the estimation of landmarks in FastSLAM algorithm. In fact, each particle carries $M$ Extended Kalman Filters that estimate the location of $M$ landmarks based on path estimation of the robot. At this step, the independency of landmarks observations makes $M$ independent estimation of landmarks based on the estimated trajectory. If a landmark is not observed at the current time step, the posterior data at the previous time step will be substituted according to the following equation:

$$
P (m_i \mid X^R_k, Z_k, U_k, x^R_0) = P (m_i \mid X^R_{k-1}, Z_{k-1}, U_{k-1}, x^R_0)
$$

(B.12)

If a landmark is observed, then the following simplified equation can estimate the location of $i^{th}$ landmark in accordance with Bayes rule and Markov localization.

$$
P (m_i \mid X^R_k, Z_k, U_k, x^R_0) P (z_k \mid X^R_k, Z_{k-1}, U_k) = P (z_k \mid m_i, x^R_k) P (m_i \mid X^R_{k-1}, Z_{k-1}, U_{k-1}, x^R_0)
$$

(B.13)

Equation (3.56) is called the landmark update equation. To better express this equation with the consideration of the non-linearity of the measurement expressed as

$$
P (z_k \mid x^R_k) \leadsto z_k = h (x^R_k, m_{ki}) + v_k
$$

(B.14)
the result will be

\[ y_k \sim N(0, R_k) \quad (B.15) \]

\[ \hat{z}_k = h (\mathbf{x}_k^R, \mu_{k-1,i}) \quad (B.16) \]

\[ \Lambda_k = \left. \frac{\partial h(\mathbf{x})}{\partial \mathbf{z}_{i,j}} \right|_{x=x_k^R, z_{i,j}=z_{k-1,i}} \quad (B.17) \]

\[ h (\mathbf{x}_k^R, z_{k,i}) \approx \hat{z}_k + \Lambda_k (z_{k,i} - \mu_{k-1,i}) \quad (B.18) \]

where \( \mu_{k-1,i} \) is the mean of \( i^{th} \) landmark at time step \( k-1 \), and \( \Lambda_k \) is the matrix of partial derivatives observation function over the observation vector of \( i^{th} \) landmark at time step \( k \).

Using equation (B.61), probability and observation density functions will be Gaussian and expressed as

\[ z_k \sim N(\hat{z}_k + \Lambda_k (z_{k,i} - \mu_{k-1,i}), R_k) \quad (B.19) \]

\[ P(\mathbf{m} | \mathbf{X}_{k-1}^R, \mathbf{Z}_{k-1}, \mathbf{U}_{k-1}, \mathbf{x}_0^R) \sim N(\mu_{k-1,i}, \Sigma_{k-1,i}) \quad (B.20) \]

The result will be the EKF algorithm which updates a landmark location at time step \( k \):

\[ \bar{Z}_k = \Lambda_k \Sigma_{k-1,i} \Lambda_k^T + R_k \quad (B.21) \]
\[ \mathbf{K}_k = n \Sigma_{k-1,i} \mathbf{A}_{k}^{T} \hat{\mathbf{Z}}_k^{-1} \]  
(B.22)

\[ n \mu_{k,i} = n \mu_{k-1,i} + \mathbf{K}_k (\mathbf{z}_k - \hat{\mathbf{z}}_k) \]  
(B.23)

\[ n \Sigma_{k,i} = [I - \mathbf{K}_k \mathbf{A}_k] n \Sigma_{k-1,i} \]  
(B.24)

where \( n \Sigma_{k,i} \) is the covariance of \( i^{th} \) landmark. When a landmark is in the range of the robot’s sensor, there is a distance \( D \) between the sensor and the landmark, and an angle \( \beta \) expressing the orientation of the landmark with respect to the sensor as shown in figure B.2.

\[ \begin{align*}
\mathbf{z}_k &= h(\mathbf{x}) + \mathbf{v}_k = \\
&\begin{bmatrix}
\mathbf{z}_{k,i}^D \\
\mathbf{z}_{k,i}^\beta
\end{bmatrix} + \\
&\begin{bmatrix}
\sqrt{(x_k^m - x_k^R)^2 + (y_k^m - y_k^R)^2} \\
\tan^{-1}\left(\frac{(y_k^m - y_k^R)}{(x_k^m - x_k^R)} - \phi_k^R + \frac{\pi}{2}\right)
\end{bmatrix} + \mathbf{v}_k
\end{align*} \]  
(B.25)
where \((x_k^{m_i}, y_k^{m_i})\) is the coordinate of the observed landmark with respect to the global reference system and \(\phi\) is the angle that the coordinate system of the sensor/robot makes with the global reference system (the sensor coordinate system is attached to the robot). The Jacobian \(\Lambda_k\) is then expressed as

\[
\Lambda_k = \begin{bmatrix}
\frac{x_k^{m_i} - x_k^R}{\Delta} & \frac{y_k^{m_i} - y_k^R}{\Delta} \\
\frac{\sqrt{\Delta}}{\Delta} & \frac{\sqrt{\Delta}}{\Delta} \\
\frac{y_k^{m_i} - y_k^R}{\Delta} & \frac{x_k^{m_i} - x_k^R}{\Delta}
\end{bmatrix}
\]  \hspace{1cm} (B.26)

\[
\Delta = (x_k^{m_i} - x_k^R)^2 + (y_k^{m_i} - y_k^R)^2
\]  \hspace{1cm} (B.27)

### B.3.3 Importance Weights

Since particles that are used to predict the path of the robot are distributed as equation (B.9), they need to be matched with

\[
P(X_k^R | Z_k, U_k, x_0^R)
\]  \hspace{1cm} (B.28)

By so called importance sampling, this matching can be done straightforward. If it is not possible to take direct sample out of a function, the importance sampling technique can be used to draw samples from a proposal function. A weight is given to each sample that is as follows:

\[
\hat{w}_k = \frac{\text{Target distribution}}{\text{Proposal distribution}} = \frac{P(aX_k^R | Z_k, U_k, x_0^R)}{P(aX_k^R | Z_{k-1}, U_k, x_0^R)}
\]  \hspace{1cm} (B.29)
This process which is called SIR algorithm is “an example of Rubin’s sampling importance resampling” [31]. According to this process, a new set of un-weighted samples is drawn from the weighted set with probabilities in proportion to the weight. Using Bayes rule and Markov assumption, the weighted samples can be expressed as

$$^n\hat{w}_k = P(Z_k | ^nX_k^R, Z_{k-1}, U_k, x_0^R)$$  \hspace{1cm} (B.30)

As the landmark estimator is done through EKF, the observation likelihood can be computed in a closed form. This probability can be calculated using the innovation and its covariance matrix. The importance weight can be then written as
\[
^{n}\hat{w}_{k} = \frac{1}{\sqrt{2\pi \tilde{Z}_{k,j}}} \exp(-\frac{1}{2} \tilde{z}_{k,j}^T \tilde{Z}_{k,j}^{-1} \tilde{z}_{k,j})
\] (B.31)

Figure B.3 depicts the drawing of samples from the proposal distribution (dashed curve) in one dimension. The solid curve illustrates the target distribution [8]. This figure shows how hard it is to draw samples from the target distribution and as a result, samples that have higher weights in the regions of the proposal distribution are larger than the target distribution and vice versa.

### B.3.4 Importance Sampling Update

Subsequent to weighting temporary particles, a new set of samples \( X_k \) is drawn with replacement from this set with probabilities in proportion to the weights. This step is called importance re-sampling in which FastSLAM corrects the pose sample \( X_k \) since there has not been any recent measurement involved in the process. Different re-sampling techniques can be found in [63].
Appendix C

Rao-Blackwellised Particle Filtering Algorithm [8]

\[ \mathbf{x}_k = \mathbf{x}_{aux} = 0 \]

for \( n = 1 \) to \( \mathcal{B} \) // loop over all particles
retrieve \( n \)-th particle \{ \[ \mathbf{x}_k^R \], \[ \mathbf{M}_{k-1} \], \[ \mathbf{\mu}_{k-1,1} \], \[ \mathbf{\sigma}_{k-1,1} \] \} from \( \mathbf{x}_{k-1} \)
draw \[ \mathbf{x}_k^R \sim P ( \mathbf{x}_k^R | \mathbf{u}_k, \mathbf{x}_{k-1} ) \] // Sample new pose
for \( i = 1 \) to \( \mathcal{M}_{k-1} \) // loop over potential data association
\[ \Lambda_k = \frac{\partial h(\mathbf{x})}{\partial \mathbf{z}_{k,i}} \bigg|_{\mathbf{x} = \mathbf{x}_k^R, \mathbf{z}_{k,i} = \mathbf{\mu}_{k-1,i}} \]
\[ \hat{\mathbf{z}}_k = h ( \mathbf{x}_k^R, \mathbf{\mu}_{k-1,i} ) \]
\[ \tilde{\mathbf{Z}}_k = \Lambda_k^T \mathbf{\Sigma}_{k-1,i} \Lambda_k^T + \mathbf{R}_k \]
\[ p_{k,i} = \frac{1}{\sqrt{2\pi|\tilde{\mathbf{Z}}_{k,i}|}} \exp(-\frac{1}{2} (\mathbf{z}_{k,i} - \hat{\mathbf{z}}_{k,i})^T \tilde{\mathbf{Z}}_{k,i}^{-1} (\mathbf{z}_{k,i} - \hat{\mathbf{z}}_{k,i})) \]
end for
\[ p_{*_{M_{k-1}}} = p_0 \]
\[ \hat{d}_{k,i} = \text{argmax} \ n P_{k,i} \text{ or draw random } \hat{d}_{k,i} \text{ with probability } \propto n P_{k,i} \] // pick a data association
if \( \hat{d}_{k,i} = \mathcal{M}_{k-1} + 1 \)
\[ \mathcal{N}_k = \mathcal{M}_{k-1} + 1 \]
\[ \mathbf{\mu}_{k,\hat{d}_k} = h^{-1} ( \mathbf{x}_k^R, \hat{\mathbf{Z}}_{k,\hat{d}_k} ) \]
\[ \mathbf{\Sigma}_{k,\hat{d}_k} = (\Lambda_{k,\hat{d}_k}^T \mathbf{R}_k^{-1} \Lambda_{k,\hat{d}_k})^{-1} \] // or is a known feature?
else
\[ \mathcal{N}_k = \mathcal{M}_{k-1} \]
\[ \mathbf{K}_{k,\hat{d}_k} = \mathbf{\Sigma}_{k-1,\hat{d}_k} \Lambda_{k,\hat{d}_k}^T \tilde{\mathbf{Z}}_{k,\hat{d}_k}^{-1} \]
\[ \mathbf{\mu}_{k,\hat{d}_k} = \mathbf{\mu}_{k-1,\hat{d}_k} + \mathbf{K}_{k,\hat{d}_k} (\mathbf{z}_k - \hat{\mathbf{z}}_{k,\hat{d}_k}) \]
\[ \mathbf{\Sigma}_{k,\hat{d}_k} = [I - \mathbf{K}_{k,\hat{d}_k} \Lambda_{k,\hat{d}_k}] \mathbf{\Sigma}_{k-1,\hat{d}_k} \]
end if
for \( i = 1 \) to \( \mathcal{M}_k \) // handle unobserved features
if \( d_k \neq \hat{d}_k \)
\[ \mathbf{\mu}_{k,m_i} = \mathbf{\mu}_{k-1,m_i} \]
\[ \sum_{k,m_i} = \sum_{k-1.m_i} \]
end if
end for

\[ n \hat{w}_k = n \hat{p}_{k,d_k} \]

add \{ n x^n_k, n M_k, n \mu_{k,1}, \ldots, \mu_{k,M_k}, \sigma_{k,1}, \ldots, \sigma_{k,M_k} \} to \textbf{x} \text{aux} \quad // \text{save weighted particle}
end for

for \( n=1 \) to \( \Phi \) \quad // \text{resample } \Phi \text{ new particles}
draw random particle from \( \text{x}_{\text{aux}} \) with probability \( \propto n \hat{w}_k \)
add new particles to \( \text{x}_k \)
end for
return \( \text{x}_k \)
Appendix D

Time and Measurement Update Process [69]

Initialization:

\[ \hat{x}_0 = E[x_0] \]
\[ \hat{P}_{x_k} = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] \]
\[ \bar{w}_k = E[w_k] \]
\[ Q_{k} = E[(w_k - \bar{w}_k)(w_k - \bar{w}_k)^T] \]
\[ \bar{v}_k = E[v_k] \]
\[ R_k = E[(v_k - \bar{v}_k)(v_k - \bar{v}_k)^T] \]

For \( k = 1 \ldots \infty \)

1. The covariance square-root column vectors for time –update:

\[ s^{x,i}_{k-1} = \ell \left( \sqrt{P_{x_{k-1}}} \right) \quad i = 1, 2, \ldots, L_x \]
\[ s^{w,j}_{k-1} = \ell \left( \sqrt{Q_k} \right) \quad i = 1, 2, \ldots, L_w \]

2. Time-update equations:

\[ \hat{x}_k = \frac{\ell^2 - L_x - L_w}{\ell^2} f(\hat{x}_{k-1}, \bar{w}_k, \bar{u}_{k-1}) \]
\[ + \frac{1}{2 \ell^2} \sum_{i=1}^{L_x} [f(\hat{x}_{k-1} + s^{x,i}_{k-1}, \bar{w}_k, \bar{u}_{k-1}) + f(\hat{x}_{k-1} - s^{x,i}_{k-1}, \bar{w}_k, \bar{u}_{k-1})] \]
\[ + \frac{1}{2\ell^2} \sum_{i=1}^{L_x} \left[ f\left( \hat{x}_{k-1} + s_{k-1}^{x_i}, \hat{x}_{k-1} - s_{k-1}^{x_i} \right) + f\left( \hat{x}_{k-1}, \hat{x}_{k-1} - s_{k-1}^{x_i} \right) \right] \]

\[ \mathbf{P}_{x_k} = \frac{1}{4\ell^2} \sum_{i=1}^{L_x} \left[ f\left( \hat{x}_{k-1} + s_{k-1}^{x_i}, \hat{x}_{k-1} - s_{k-1}^{x_i} \right) \right]^2 \]

\[ + \frac{1}{4\ell^2} \sum_{i=1}^{L_x} \left[ f\left( \hat{x}_{k-1}, \hat{x}_{k-1} - s_{k-1}^{x_i} \right) \right]^2 \]

3. Calculate covariance square-root column vectors for measurement-update:

\[ \mathbf{s}_{k-1}^{x_i} = \ell \left( \sqrt{\mathbf{P}_{x_k}^{-1}} \right) \quad i = 1, 2, \ldots, L_x \]

\[ \mathbf{s}_{k-1}^{x_i} = \ell \left( \sqrt{\mathbf{R}_{k}^{-1}} \right) \quad i = 1, 2, \ldots, L_v \]

4. Measurement-update equations:

\[ \hat{z}_k = \frac{\ell^2 - L_x - L_v}{\ell^2} h(\hat{x}_k, \hat{v}_k) \]

\[ + \frac{1}{2\ell^2} \sum_{i=1}^{L_x} \left[ h(\hat{x}_k + s_{k}^{x_i}, \hat{v}_k) + h(\hat{x}_k - s_{k}^{x_i}, \hat{v}_k) \right] \]

\[ + \frac{1}{2\ell^2} \sum_{i=1}^{L_v} \left[ h(\hat{x}_k + s_{k}^{x_i}, \hat{v}_k) + h(\hat{x}_k, \hat{v}_k - s_{k}^{x_i}) \right] \]

\[ \mathbf{P}_{y_k} = \frac{1}{4\ell^2} \sum_{i=1}^{L_x} \left[ h(\hat{x}_k + s_{k}^{x_i}, \hat{v}_k) - h(\hat{x}_k + s_{k}^{x_i}, \hat{v}_k) \right]^2 \]
\[ + \frac{1}{4\ell^2} \sum_{i=1}^{L_x} \left[ h(\hat{x}_k^i, \bar{v}_k) - h(\hat{x}_k^i, \bar{v}_k) + s_k^{xi} - s_k^{yi} \right]^2 \]

\[ + \frac{\ell^2 - 1}{4\ell^2} \sum_{i=1}^{L_x} \left[ h(\hat{x}_k^i, \bar{v}_k) + h(\hat{x}_k^i, \bar{v}_k) - 2h(\hat{x}_k^i, \bar{v}_k) \right]^2 \]

\[ + \frac{\ell^2 - 1}{4\ell^2} \sum_{i=1}^{L_x} \left[ h(\hat{x}_k^i, \bar{v}_k) + h(\hat{x}_k^i, \bar{v}_k) - 2h(\hat{x}_k^i, \bar{v}_k) \right]^2 \]

\[ P_{x_k y_k} = \frac{1}{4\ell^2} \sum_{i=1}^{L_x} s_k^{xi} \left[ h(\hat{x}_k^i, \bar{v}_k) - h(\hat{x}_k^i, \bar{v}_k) \right]^{T} \]

\[ K_k = P_{x_k y_k} P_{y_k}^{-1} \]

\[ \hat{x}_k = \hat{x}_k^i + K_k (y_k - \bar{y}_k) \]

\[ P_{x_k} = P_{x_k} - K_k P_{x_k} K_k^{T} \]

Parameters $\ell \geq 1$ is the scalar central difference interval size. For Gaussian $x$, the optimal value is $\ell = \sqrt{3}$. $L_x$, $L_w$, and $L_v$ are the dimensions of the state, process noise and observation noise respectively. $Q_k$ is the covariance matrix of motion noise and $R_k$ is the covariance matrix of the observation noise. $(.)^2$ is the shorthand for the vector outer product, i.e. $a^2 = a a$, and $\left( \sqrt{P} \right)_i$ is the $i$th column of the matrix square root of the square-symmetric matrix $P$. 
## Appendix E

### Technical Features of the Processor used for Simulations

<table>
<thead>
<tr>
<th>Processor / Chipset</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Core 2 Duo T9400 / 2.53 GHz</td>
</tr>
<tr>
<td>Number of Cores</td>
<td>Dual-Core</td>
</tr>
<tr>
<td>Cache</td>
<td>L2 cache - 6.0 MB</td>
</tr>
<tr>
<td>64-bit Computing</td>
<td>Yes</td>
</tr>
<tr>
<td>Front Side Bus</td>
<td>1066.0 MHz</td>
</tr>
<tr>
<td>Chipset</td>
<td>Mobile Intel GM45 Express</td>
</tr>
<tr>
<td>Platform Technology</td>
<td>Intel Centrino 2 with vPro Technology</td>
</tr>
<tr>
<td>Features</td>
<td>Intel Dynamic Acceleration, Intel Trusted Execution Technology, Intel 64 Technology, Intel Virtualization Technology</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ram</td>
<td>2.0 GB ( 1 x 2 GB )</td>
</tr>
<tr>
<td>Max Ram Supported</td>
<td>4.0 GB DDR3 SDRAM</td>
</tr>
<tr>
<td>Speed</td>
<td>1066.0 MHz / PC3-8500</td>
</tr>
<tr>
<td>Form Factor</td>
<td>SO DIMM 204-pin</td>
</tr>
</tbody>
</table>