Contributions to Mobile Robot Localisation in GPS-Denied Environments

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Robin Ping Guan

Master of Engineering (Electrical) – The University of Melbourne
Bachelor of Commerce (Economics, Finance) – The University of Melbourne

School of Engineering
College of Science, Engineering and Health
RMIT University

August 2019
Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Robin Ping Guan

October 21, 2019
Dedication

To my mother and my father.
Acknowledgements

I must extend my utmost appreciation for my supervisors and acknowledge their contributions. They have contributed their ideas, time, and care, which played a huge part in completing this thesis.

As my primary supervisor, Dr. Branko Ristic helped me with his guidance, dedication towards research, and his ability to define specific goals and targets when completing a project. He set a great example academically but also, in leadership.

I must also thank Professor Liuping Wang as my second supervisor for her help and support in leadership, and administration. She helped with keeping projects running smoothly, and often provided the final push to conclude a project.

Furthermore, I would like to acknowledge the following people for their assistance in relation to the work in Chapter 5.5.2:

Ms. Rachel Geddes, and Ms. Margo Brown helped to bring this project to me. I have benefited greatly from this experience, and I thank them for helping to connect me with this opportunity.

Dr. Jennifer L. Palmer provided the leadership, and coordination for her team working on hardware, and experiments. In this team, Mr. Ricardo Cannizzaro, and Mr. Lance Fang helped to collect the experimental data. I would like to thank these three people for their contribution to this thesis.

Thank you to Defence Science Institute (DSI) for a scholarship top-up.

I would also like to express my gratitude to the following people in the RMIT community for their friendship and support: Mr. Xingda Chen, Mr. Long Tran Quang, Mr. Yuankang Zhu, Dr. Yangbo Liu, Dr. Lu Gan, Dr. Xi Chen and Dr. Pakorn Poksawat. Thank you to a number of other friends from outside of RMIT that I have been in touch with in the past few years: Dr. Tien Foong, Mr. Christian Suprijatna, Mr. Vernon Hu and Ms. Sophia Bian. Finally, thank you to my friends from badminton: Mr. Ivan Kuan,
Mr. Louie Saw, Ms. Josie Fan, Mr. Tiklik Lee and Mr. Very Lim.
# Contents

1 Introduction ............................................................................. 3
   1.1 Overview ........................................................................ 3
   1.2 Structure of the Thesis .................................................. 5
   1.3 Contributions ............................................................... 6
   1.4 Publications ................................................................... 6
      1.4.1 Journal Publications ............................................. 6
      1.4.2 Conference Publications ...................................... 7

2 Background and Literature Review .......................................... 9
   2.1 Introduction .................................................................... 9
   2.2 Mobile Robot Localisation in GPS-Denied Environments .... 9
   2.3 Mobile Robot Localisation Scenarios ............................... 10
   2.4 Sensors for Mobile Robots ............................................ 11
   2.5 Algorithms for Mobile Robot Localisation ...................... 16
      2.5.1 Kalman filters ................................................... 17
      2.5.2 Particle filters .................................................. 19
   2.6 Computation ............................................................... 30
   2.7 Conclusion ................................................................. 32

3 Localisation using Doppler Radar ......................................... 33
   3.1 Introduction ............................................................... 33
   3.2 Motion Model ............................................................ 34
      3.2.1 Velocity Motion Model ........................................ 34
   3.3 Measurement Model .................................................... 36
   3.4 Cramer-Rao Lower Bound (CRLB) ................................. 39
   3.5 Extended Kalman Filter (EKF) ...................................... 41
      3.5.1 Simulation Results ............................................. 43
      3.5.2 Simulation Results: Varying Landmarks ................. 55
## CONTENTS

3.6 Sequential Importance Resampling (SIR) Particle Filter . . . . 60
3.7 Exact Daum-Huang (EDH) Particle Flow Particle Filter . . . 63
3.7.1 Overview of the EDH Particle Flow Particle Filter . . . 64
3.7.2 Implementation . . . . . . . . . . . . . . . . . . . . . . 67
3.7.3 Simulation Results and Comparative Studies . . . . . . 71
3.8 Conclusion and Discussion . . . . . . . . . . . . . . . . . . . . 81

4 Localisation using Doppler Radar: Relaxing the Assump-
tions 83
4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
4.2 Motion Model . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
4.3 Measurement Model . . . . . . . . . . . . . . . . . . . . . . . 85
4.3.1 Association Hypotheses . . . . . . . . . . . . . . . . . . . 87
4.3.2 Likelihood Function . . . . . . . . . . . . . . . . . . . . . 88
4.3.3 Murty’s Algorithm . . . . . . . . . . . . . . . . . . . . . 90
4.4 RFS Particle Filter with KLD-Sampling . . . . . . . . . . . . 92
4.4.1 KLD-Sampling . . . . . . . . . . . . . . . . . . . . . . . 92
4.4.2 Implementation of the RFS Particle filter with KLD-
Sampling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
4.4.3 Simulation Results . . . . . . . . . . . . . . . . . . . . . . 97
4.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 104

5 KLD-Gmapping 105
5.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . 105
5.2 Gmapping Proposal Distribution . . . . . . . . . . . . . . . . 106
5.3 Inertial Navigation-Based Motion Model . . . . . . . . . . . . 110
5.4 LIDAR measurement model . . . . . . . . . . . . . . . . . . . 111
5.4.1 Laser beam successfully hits an object in the map and
returns a measurement . . . . . . . . . . . . . . . . . . . . . . 113
5.4.2 Laser beam was blocked by an unexpected object . . . 113
5.4.3 No obstacles were detected within range . . . . . . . . . 114
5.4.4 Laser returns were random . . . . . . . . . . . . . . . . 114
5.5 KLD-Gmapping . . . . . . . . . . . . . . . . . . . . . . . . . . 115
5.5.1 KLD-Gmapping Simulation Results . . . . . . . . . . . . 118
5.5.2 KLD-Gmapping Experiment . . . . . . . . . . . . . . . . 125
5.6 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 133

6 Conclusion and Future Work 135
CONTENTS
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Illustration of mobile robot localisation with Doppler radar</td>
<td>38</td>
</tr>
<tr>
<td>3.2</td>
<td>A single realisation of the EKF compared to the velocity motion model</td>
<td>45</td>
</tr>
<tr>
<td>3.3</td>
<td>Cramer-Rao lower bound (CRLB) for the number of landmarks $M = 1, 2, 3, 5, 7$</td>
<td>46</td>
</tr>
<tr>
<td>3.4</td>
<td>RMSE for the number of landmarks $M = 1, 2, 3, 5, 7$</td>
<td>47</td>
</tr>
<tr>
<td>3.5</td>
<td>Comparisons between the CRLB, EKF RMSE, and velocity motion model RMSE for the case $M = 1$ landmark</td>
<td>48</td>
</tr>
<tr>
<td>3.6</td>
<td>Comparisons between the CRLB, EKF RMSE, and velocity motion model RMSE for the case $M = 2$ landmark</td>
<td>49</td>
</tr>
<tr>
<td>3.7</td>
<td>Comparisons between the CRLB, EKF RMSE, and velocity motion model RMSE for the case $M = 3$ landmark</td>
<td>50</td>
</tr>
<tr>
<td>3.8</td>
<td>Comparisons between the CRLB, EKF RMSE, and velocity motion model RMSE for the case $M = 5$ landmark</td>
<td>51</td>
</tr>
<tr>
<td>3.9</td>
<td>Comparisons between the CRLB, EKF RMSE, and velocity motion model RMSE for the case $M = 7$ landmark</td>
<td>52</td>
</tr>
<tr>
<td>3.10</td>
<td>Cramer-Rao lower bound (CRLB) for variations of uncertainty on the Doppler-shift measurement. There are $M = 7$ landmarks</td>
<td>53</td>
</tr>
<tr>
<td>3.11</td>
<td>Cramer-Rao lower bound (CRLB) for variations of uncertainty on the azimuth measurement. There are $M = 7$ landmarks</td>
<td>54</td>
</tr>
<tr>
<td>3.12</td>
<td>A single realisation of the EKF with $M = 7$ landmarks</td>
<td>57</td>
</tr>
<tr>
<td>3.13</td>
<td>CRLB at the final discrete-time index $k = 800$ for a varying number of landmarks</td>
<td>58</td>
</tr>
<tr>
<td>3.14</td>
<td>RMSE at the final discrete-time index $k = 800$ for a varying number of landmarks</td>
<td>59</td>
</tr>
<tr>
<td>3.15</td>
<td>A single realisation for $M = 3$ landmarks</td>
<td>73</td>
</tr>
<tr>
<td>3.16</td>
<td>RMSE for $M = 3$ landmarks</td>
<td>74</td>
</tr>
<tr>
<td>3.17</td>
<td>A single realisation for $M = 4$ landmarks</td>
<td>76</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>3.18</td>
<td>RMSE for $M = 4$ landmarks. 77</td>
<td></td>
</tr>
<tr>
<td>3.19</td>
<td>A single realisation for $M = 5$ landmarks. 79</td>
<td></td>
</tr>
<tr>
<td>3.20</td>
<td>RMSE for $M = 5$ landmarks. 80</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>A single realisation of the RFS particle filter. 99</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Monte Carlo results for scenario C. 103</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Illustration of the KLD-Gmapping scenario. 112</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>Example PDF of a single LIDAR beam measurement in simulation. 119</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>A single realisation of two simulated scenarios comparing KLD localisation against KLD-Gmapping. 120</td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>Comparison of the RMSEs between KLD localisation and KLD-Gmapping for Scenario One. 122</td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>Comparison of the RMSEs between KLD localisation and KLD-Gmapping for Scenario Two. 123</td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>Comparison of the number of particles between KLD localisation and KLD-Gmapping for Scenario One. 124</td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>Comparison of the number of particles between KLD localisation and KLD-Gmapping for Scenario Two. 124</td>
<td></td>
</tr>
<tr>
<td>5.8</td>
<td>Actual measurements received (green lines), at $k = 10$ from the ground-truth trajectory (blue) during the experiment. 127</td>
<td></td>
</tr>
<tr>
<td>5.9</td>
<td>Example PDF of a single LIDAR beam applied to the experimental results. 128</td>
<td></td>
</tr>
<tr>
<td>5.10</td>
<td>A single realisation of KLD-Gmapping compared against the ground truth, KLD localisation and the trajectory from the simulated inertial navigation data. 129</td>
<td></td>
</tr>
<tr>
<td>5.11</td>
<td>Comparison of the number of particles between KLD localisation and KLD-Gmapping for the experimental results. 131</td>
<td></td>
</tr>
<tr>
<td>5.12</td>
<td>Comparison of the RMSEs of KLD-Gmapping, KLD localisation and the simulated inertial navigation data. 132</td>
<td></td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Scenario with three landmarks.</td>
<td>75</td>
</tr>
<tr>
<td>3.2</td>
<td>Scenario with four landmarks.</td>
<td>78</td>
</tr>
<tr>
<td>3.3</td>
<td>Scenario with five landmarks.</td>
<td>81</td>
</tr>
<tr>
<td>4.1</td>
<td>Association function in relation to hypotheses and landmarks</td>
<td>87</td>
</tr>
<tr>
<td>4.2</td>
<td>Full list of hypotheses for $M = 2, L_k = 3$.</td>
<td>88</td>
</tr>
<tr>
<td>4.3</td>
<td>RFS PF simulation results under various assumptions.</td>
<td>101</td>
</tr>
<tr>
<td>5.1</td>
<td>KLD parameters for the simulation.</td>
<td>118</td>
</tr>
<tr>
<td>5.2</td>
<td>Computation time (in seconds) for KLD localisation and KLD-Gmapping in simulation.</td>
<td>125</td>
</tr>
<tr>
<td>5.3</td>
<td>KLD parameters for the experimental data.</td>
<td>130</td>
</tr>
<tr>
<td>5.4</td>
<td>Computation time (in seconds) for the experimental application of KLD localisation and KLD-Gmapping.</td>
<td>131</td>
</tr>
</tbody>
</table>
LIST OF TABLES
### List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d^{(j)}_k$</td>
<td>Doppler shift caused by landmark $j$ at discrete-time index $k$</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>sampling interval</td>
</tr>
<tr>
<td>$k$</td>
<td>discrete-time index</td>
</tr>
<tr>
<td>$L_k$</td>
<td>number of received measurements at discrete-time index $k$</td>
</tr>
<tr>
<td>$m$</td>
<td>map, which is assumed to be known and time-invariant</td>
</tr>
<tr>
<td>$m^{(j)}$</td>
<td>position of landmark $j$ in the map</td>
</tr>
<tr>
<td>$M$</td>
<td>number of landmarks in the map</td>
</tr>
<tr>
<td>$N$</td>
<td>number of particles</td>
</tr>
<tr>
<td>$N_{\text{KLD}}$</td>
<td>KLD number of particles</td>
</tr>
<tr>
<td>$x_k$</td>
<td>robot’s pose $[x \ (m), \ y \ (m), \ \theta \ (\text{rad})]^T$ at discrete-time index $k$</td>
</tr>
<tr>
<td>$x^{[i]}_k$</td>
<td>the pose of particle $i$ at discrete-time index $k$</td>
</tr>
<tr>
<td>$\phi^{(j)}_k$</td>
<td>azimuth reading from the robot’s pose to landmark $j$ at discrete-time index $k$</td>
</tr>
<tr>
<td>$\pi(x_k</td>
<td>x_{k-1}, u_k)$</td>
</tr>
<tr>
<td>$\psi(Z_k</td>
<td>x_k, m, u_k)$</td>
</tr>
<tr>
<td>$\psi(Z_k</td>
<td>x_k, m)$</td>
</tr>
<tr>
<td>$Q_k$</td>
<td>noise covariance for the robot’s motion model at discrete-time index $k$</td>
</tr>
<tr>
<td>$R_k$</td>
<td>noise covariance for the robot’s measurement model at discrete-time index $k$</td>
</tr>
<tr>
<td>$T$</td>
<td>number of Monte Carlo trials</td>
</tr>
<tr>
<td>$u_k$</td>
<td>robot’s input control signal at discrete-time index $k$</td>
</tr>
<tr>
<td>$w^{[i]}_k$</td>
<td>normalised weight of particle $i$ at discrete-time index $k$</td>
</tr>
<tr>
<td>$\tilde{w}^{[i]}_k$</td>
<td>unnormalised weight of particle $i$ at discrete-time index $k$</td>
</tr>
<tr>
<td>$z^{(j)}_k$</td>
<td>measurement $j$ received at discrete-time index $k$</td>
</tr>
<tr>
<td>$Z_k$</td>
<td>the set of all measurements received at discrete-time index $k$</td>
</tr>
</tbody>
</table>
List of Key Acronyms

CRLB  Cramer-Rao lower bound
EKF   extended Kalman filter
ESS   effective sample size
GPS   Global Positioning System
GNN   global nearest neighbour
IMU   inertial measurement unit
KLD   Kullback-Leibler distance
LIDAR light detection and ranging
MCL   Monte Carlo localisation
ML    maximum likelihood
MMSE  minimum mean square error
PF    particle filter
RFS   random finite set
RMSE  root mean square error
SIR   sequential importance resampling
SIS   sequential importance sampling
SLAM  simultaneous localisation and mapping
Abstract

Localisation of mobile robots is an important area of research where the aim is to produce mobile robots that can assist with automating human activities in environments without GPS. These GPS-denied environments could be indoor, underground, undersea, or extra-terrestrial. Mobile robot localisation is most commonly performed using cameras, and LIDAR/sonar range sensors. The novel possibility of achieving the same goal with Doppler radars could produce robots that are smaller, cheaper and use less power.

This thesis presents the following contributions: First, it demonstrates that mobile robot localisation using a Doppler radar in a known, feature-based map is possible under basic assumptions such as: information of the initial pose, Gaussian measurement noise, and known landmark-measurement associations. The extended Kalman filter, particle filter and state-of-the-art Exact Daum-Huang particle flow particle filter were applied. The feasibility is studied using the Cramer-Rao lower bound.

Second, the thesis demonstrates that mobile robot localisation using a Doppler radar in a known, feature-based map is possible under more realistic assumptions. There is no longer any information about the robot’s initial pose, and the radar noise is no longer Gaussian - it is possible for the radar to produce missed, and false detections. Landmark-measurement associations are now unknown. The solution is based on the random finite set particle filter, combined with an adaptive sample size algorithm to reduce computation. The number of hypothesis evaluated by the particle filter is limited by use of Murty’s algorithm, also for the purpose of reducing computation.

Third, the thesis combines particle filters with intelligent proposal distributions, with an adaptive sample size algorithm. The resulting algorithm has been proven to be effective in mobile robot localisation applications in both, simulated and experimental datasets using a LIDAR sensor. The resulting algorithm has a greater accuracy with a lower number of particles compared
to a benchmark algorithm.

Keywords: Localisation, Kalman Filter, Particle Filter, Doppler Radar, Mobile Robots, Bayesian Estimation, Autonomous Robot Exploration.
Chapter 1

Introduction

1.1 Overview

GPS is unavailable in a number of environments, for example: indoor, underground, underwater, or extra-terrestrial environments. Work may need to be performed in these GPS-denied environments. These tasks can expose humans to danger e.g. underwater or space exploration, or can be extremely repetitive e.g. certain warehouse or indoor cleaning tasks. In order to outsource these tasks to a robot, it is often a prerequisite that the robot must know its own position and heading in the GPS-denied environment. The mobile robot localisation problem consists of estimating the “pose” of a mobile robot using non-GPS sensors with a given map of the environment. The pose at discrete-time index $k$, is a vector: $\mathbf{x}_k = [x_k, y_k, \theta_k]^T$, consisting of Cartesian coordinates of the robot’s position: $x_k$ (meters), $y_k$ (meters) and its heading angle $\theta_k$ (radians). Cox [1, p. 193] describes the problem of mobile robot localisation as: “a primary problem that must be solved for autonomous vehicles working in structured environments”. Cox [1, p. 196] further states the importance of this problem: “Using sensory information to locate the robot in its environment is the most fundamental problem to providing a robot with autonomous capabilities”.

An extension to this problem is called: Simultaneous Localisation and Mapping (SLAM) where the robot must estimate its own pose as well as the map which is unknown. SLAM is not explicitly studied in this thesis, but is occasionally referenced as it is an extension of the mobile robot localisation problem.
Mobile robot localisation requires an “exteroceptive” (relating to external stimuli) sensor(s). Popular exteroceptive sensors for modern robots include ranging sensors such as LIDARs and ultrasonic range sensors, and image sensors such as cameras. Traditional sensors for localisation, such as ultrasonic sensors, have declined in popularity \[2, p. 1324\], whereas some newer hybrid sensors such as the Microsoft Kinect (which has the ability to capture images like a camera and measure distance like a LIDAR) have gained interest in recent robotics applications. This thesis investigates two types of sensor modalities: LIDAR, and Doppler radar. LIDARs have previously been used to solve the mobile robot localisation problem, whereas Doppler radars have not. Doppler radars are a key area of research of this thesis for reasons such as being: “small, light, simple in structure, and cheap” \[3, p. 138\] and “low-cost, low-power” \[4\].

Mobile robot localisation assumes that a map of the environment is known. This is a reasonable assumption for environments that have already been explored. Day-to-day examples could be houses, warehouses, offices, mines etc. This thesis investigates two types of maps: feature-based, and occupancy grid-based maps. Feature-based maps are sparse, mostly empty space, and occasionally have discrete landmarks which are able to return a signal. This type of map is good for representing wide, open spaces with occasional points of interest such as a large, empty warehouse with a few detectable items scattered throughout. Occupancy grid-based maps represent an environment with a grid and each grid cell can be either occupied or unoccupied. Occupancy grid-based maps are good for representing enclosed, dense spaces with many fine details, for example, an office.

The robot also uses a source of “proprioceptive” (relating to self-positioning and movement) data known as the “control input”. The type of control input data available depends on the robot’s equipment. Two types of control input signals have been investigated: the “velocity motion model”, which is based on the rotational and translation velocity signals sent to the robot \[5, Chapter 5.3\], and inertial navigation-based data that could be obtained from an inertial measurement unit (IMU). Using this information, the movement of the robot can be crudely inferred. This inference will be used to supplement the exteroceptive sensor data (from the LIDAR or the Doppler radar) to provide an estimate of the robot’s pose using signal processing algorithms.

The data from the exteroceptive sensor, proprioceptive sensor, and the known map are combined (or “fused”) together by an algorithm to produce an estimate of the robot’s pose. Many of the popular algorithms for mobile
robot localisation are based on Bayesian filtering. The thesis focuses on application of two popular Bayesian filtering methods, the extended Kalman filter (EKF), and particle filters, to solve the mobile robot localisation problem. Regarding particle filters, the thesis covers topics such as reducing computation by adapting the number of particles and the design of intelligent proposal distributions, which are key requirements for efficient implementation.

1.2 Structure of the Thesis

The structure of the thesis is presented below:

1. Chapter 2 contains the background and literature review covering a number of topics pertaining to this thesis on mobile robot localisation including: types of maps, scenarios, sensors, algorithms, proposal distributions and computation speed.

2. Chapter 3 presents the application of the extended Kalman filter, SIR particle filter, and the Exact Daum-Huang particle flow particle filter to solve the mobile robot localisation problem using a Doppler radar under a set of restrictive assumptions and various simulation settings. This chapter also contains a study of the Cramer-Rao lower bound (CRLB) to investigate the feasibility of using a Doppler radar for localisation.

3. Chapter 4 greatly relaxes the assumptions used in Chapter 3 for a more realistic situation with unknown initial pose distribution, unknown landmark-measurement associations, missed detections and false detections. The solution to this more realistic problem is based on the random finite set particle filter. The solution is combined with KLD-sampling to reduce the computation time.

4. Chapter 5 combines two well-known algorithms: KLD-sampling to reduce computation time, and the “Gmapping” proposal distribution for the particle filter. This chapter shows that the two algorithms can successfully be combined and this is demonstrated in simulation, as well as, with experimental data.

5. Chapter 6 contains the conclusions of this thesis.
1.3 Contributions

1. Chapter 3 demonstrates that mobile robot localisation using a Doppler radar in a known, feature-based map is possible in theory and in simulation. The feasibility is supported with an analysis of the Cramer-Rao lower bound. This chapter also demonstrates that state-of-the-art particle flow particle filters can potentially be applied to solve the mobile robot localisation problem. This chapter uses the following simplifying assumptions: known initial pose distribution, known landmark-measurement associations and Gaussian measurement noise.

2. Chapter 4 is an extension of the work presented in Chapter 3 showing that localisation using a Doppler radar in a known, feature-based map is possible under much more realistic conditions. These conditions are: no information of the initial pose, unknown landmark-measurement associations, and inclusion of missed detections and false detections. The solution is based on the random finite set particle filter. Computational resources are also reduced by inclusion of an adaptive sample size algorithm for the particle filter, as well as using Murty’s algorithm which limits the number of hypotheses evaluated by the particle filter.

3. Chapter 5 presents a novel Monte Carlo Localisation (MCL) algorithm which combines an intelligent proposal distribution with an adaptive sample size algorithm. This combination has been proven to be effective in localisation when applied to simulation as well as an experimental dataset collected with a LIDAR. The results show that the resulting algorithm is more accurate and requires less particles when compared to a benchmark algorithm.

1.4 Publications

1.4.1 Journal Publications

1.4. PUBLICATIONS

2. Robin Ping Guan, Branko Ristic, Liuping Wang, and Rob Evans. “Monte Carlo localisation of a mobile robot using a Doppler–Azimuth radar”. In: *Automatica* 97 (2018), pp. 161–166. DOI: [10.1016/j.automatica.2018.08.012](https://doi.org/10.1016/j.automatica.2018.08.012)


1.4.2 Conference Publications


Chapter 2

Background and Literature Review

2.1 Introduction

This chapter contains a literature review of research in the field of mobile robot localisation. A number of key topics are explored within this chapter: types of maps, scenarios, sensors, algorithms used for mobile robot localisation with a focus on Bayesian filtering and computational concerns.

2.2 Mobile Robot Localisation in GPS-Denied Environments

GPS is unavailable in a number of environments such as: indoor, underground, underwater, or extra-terrestrial environments. In these GPS-denied environments, it may be beneficial to assign certain tasks to a mobile robot because the task may be dangerous or repetitive for a human to perform. A prerequisite to many tasks would require the robot to estimate its own position and heading within the map of the environment. Mobile robot localisation is the solution to this problem. Mobile robot localisation assumes that the robot knows the map of the environment and is equipped with sensors which can give information about the robot’s motion, as well as perceive the environment. The map is usually one of the following common types: Feature-based maps are mostly sparse with occasional landmarks,
for example, a large, empty warehouse with a few detectable items scattered throughout. **Occupancy grids** [11], divide the environment into 2D (or 3D) cells. Each cell stores a probabilistic estimate of the cell’s state - usually empty, or occupied. For example, a map of an office can be modelled as an occupancy grid-based map with the walls and furniture represented as occupied grid cells. For localisation, the map is assumed to be known so the occupancy grid map represents the state (empty, or occupied) rather than a probability of the cell being empty or occupied. Burgard et al. [12] extend this idea to create a grid of probabilities representing the posterior of the current position of the mobile robot. More recently, Behzadian et al. [13] consider localisation based on a crude, hand-drawn map. Occupancy grid maps are detailed and therefore, can consume a lot of memory. Instead, **topological** maps describe the environment as a graph/network which can be more efficient. Significant landmarks or places are represented as nodes in the graph. Nodes are connected by arcs if there is a path between the nodes. Defining a landmark/place as a node is a challenge in building topological maps [14], [15]. Hertzberg [16] defines pipes, manholes, inlets etc. in a sewerage system as landmarks. These landmarks are detected by using a neural network to identify patterns from the ultrasound transducer, which can measure distance. The ultrasound transducer is carried by a mobile robot in the sewer. Finally, there are modern combinations of map types such as: occupancy grid maps combined with topological maps [15], [17]–[19]. These combine the advantages of both types of maps. Occupancy grid maps are easy to build, and are very detailed. However, they require more memory to maintain. Topological maps are very efficient to maintain, but do not contain as much detail.

Filliat and Meyer [20] present a survey on map types and algorithms used for mobile robot localisation.

### 2.3 Mobile Robot Localisation Scenarios

A number of different scenarios for mobile robot localisation also exist, as explained by Fox et al. [21].

First, is the initial pose of the robot approximately known? If so, the algorithms only require to keep track of the robot. However, if there is no information about the initial pose of the robot, the problem is more difficult because it requires solving a *global* localisation problem as well as tracking of
the robot pose. This case is also referred to as the \textit{wake-up} robot problem.

Second, can the robot identify a localisation failure and recover from it? This could occur if the localisation algorithm fails to converge correctly or if the robot is carried to another location. This scenario is commonly known as the \textit{kidnapped robot} problem.

Third, is the environment static or dynamic? Can the localisation algorithm cope with dynamic environments that change with time, for example, a room with moving people inside it, or a room that has doors which open and close frequently? Furthermore, some objects may form part of the map but are not detectable by the robot’s sensors, for example, cages, bars etc. Fox et al. \cite{21} produced a robot that can still localise in these dynamic environments by showing experimental results from a museum with visitors. Schulz et al. \cite{22}, \cite{23} used a robot equipped with two laser-range scanners and a particle filter to track moving people in the environment. Schulz et al. use joint probabilistic data association filters (JPDAFs) to associate the correspondence between the measurements and the object to be tracked and can deal with occlusions. Montemerlo et al. \cite{24} present the \textit{conditional} particle filter for simultaneously localising a mobile robot and the locations of people nearby, given a map of the environment. An extension to the SLAM problem is SLAM and moving object tracking (SLAMMOT). Wang et al. \cite{25} present SLAM with DATMO (detection and tracking of moving objects) as a solution to the SLAMMOT problem. Environments with static and dynamic portions are represented separately in solving the SLAM problem by Wolf and Sukhatme \cite{26}, \cite{27}.

\section{Sensors for Mobile Robots}

The use of different sensor modalities is a field of research for mobile robot localisation. The choice of sensors depends on a number of variables such as: price, weight, energy consumption, accuracy, type of measurement and environmental variables such as GPS availability, indoor/outdoor, weather and day/night usage.

The simplest class of sensors can estimate the current robot’s pose based on the previous known pose of the robot modified by some crude information of the robot’s motion. For example, given the previous pose, the current pose can be crudely inferred based on information of the robot’s velocity and elapsed time. This class of sensors is commonly referred to as “proprio-
CEPTIVE” i.e. relating to self-positioning and movement, or “dead-reckoning” sensors. Borenstein [28] provides an excellent description of this class of sensors. The first type of proprioceptive sensor is from “odometry” which uses the robot’s wheel rotational data which is then translated into the robot’s linear displacement on the floor. In principle, this involves the integration of incremental motion data over time which leads to an unbounded error as it accumulates over time. Although, it is accurate for short distances. Errors from odometry can result from systematic errors (resulting from kinematic imperfections of the robot and its modelling) as well as non-systematic errors (for example, the robot wheels are turning but the robot itself is not moving due to obstruction from an object or wheel slippage). Despite its weaknesses, odometry is the most widely used method of navigation for mobile robot positioning. The second type of proprioceptive sensor is called inertial navigation, which relies on inertial measurement unit (IMU) data such as gyroscopes and accelerometers to infer the motion of the robot. For example, the accelerometer data can be integrated twice to obtain an estimate of the change in position. Due to the integration process, this would cause the errors to be unbounded over time. There also exists the “velocity motion model” which is used throughout this thesis. This model relies on the rotational and translational control input signals which are used to actuate the robot’s motion [5, Chapter 5.3].

The other class of sensors is known as “exteroceptive” i.e. relating to external stimuli. Ultrasonic sensors were a classical exteroceptive sensor used in mobile robotic applications, see [12, 16]. The use of ultrasonic sensors for localisation has rapidly declined as a result of the production of cheap, optical range sensors such as 2-D LIDAR which helped to create very robust localisation and SLAM systems. Recently, 3-D LIDARs have been driving research into autonomous cars. Visual sensors such as cameras have also been highly researched in recent years [2]. Abrate et al. [29] has shown that with line extraction and data association procedures, reasonable SLAM results can be obtained with cheap, noisy infrared sensors. Cameras have also been applied to solve mobile robot localisation by Lenser and Veloso [30]. Vlassis et al. [31] use omnidirectional vision for robot localisation that is robust to large amounts of image occlusion and can solve the kidnapped robot problem. Schulz and Fox [32] use a camera for robot localisation that can adapt to changes in lighting conditions such as when a light is turned on or off. Wolf et al. [33] use an image-based system for localisation that is robust to occlusions and relies on features that are invariant to image translations.
2.4. SENSORS FOR MOBILE ROBOTS

and scale (up to a factor of two). This system is also able to solve the global localisation problem and can recover from localisation failures. Jeong and Lee’s CV-SLAM requires just a single ceiling vision sensor (a single camera pointing toward the ceiling) to solve the SLAM problem.

There are a number of costs of acquiring data as identified by Censi et al. The costs relevant to this thesis are identified as follows: Effort required to carry the sensor - affected by weight, size, shape etc., power requirements of the sensor and power requirements for data processing.

These points of discussion motivated the research into the idea of using Doppler radars applied to solve the mobile localisation problem. Doppler radars are “small, light, simple in structure, and cheap” and “low-cost, low-power”. Because they are light and low-powered, they would be beneficial for longer-range missions. Being cheap and less accurate, they can be used in missions that are non-critical and possibly require many robots.

Doppler radars also have a number of other well-known applications. For example, Doppler radars have been studied as part of vehicle applications. Gresham et al. have developed a Doppler radar module that can be used as part of an autonomous cruise control system. Woll describes a Doppler radar system that can track multiple targets to be used for collision warnings, collision avoidance and adaptive/intelligent cruise control. Heide et al. report on a Doppler sensor system that can measure the vehicle's ground-speed and position relative to a fixed reference point. This data can be used to improve advanced vehicle control systems such as anti-slip control and anti-lock braking systems. Doppler radars are also found in the aviation field.

Doppler radars have also seen applications in the health field. Geisheimer et al. use a continuous-wave radar and analyse the Doppler shift from moving arms, legs and torso which move at different velocities, to analyse gait. Saho et al. use simulated micro-Doppler radar signals to measure gait patterns and classify people as either a: healthy young adult, a healthy elderly adult or an elderly adult with a history of falls. Palmer et al. analyse gait patterns collected from Doppler signatures to potentially identify abnormalities such as concussions or other neurological diseases. Zakrzewski et al. use a microwave Doppler radar to detect the distance and heart rate of a person, even through thick clothing. Shahhaidar et al. use a Doppler radar to detect breathing rate, torso displacement, torso movement velocity and acceleration when a person breathes. Avagyan et al. propose a Doppler radar that can analyse human heartbeat and respiration.
Doviak and Zrnic [46] describe the applications of Doppler radar to meteorology in their textbook. For example, wind, precipitation, storms etc. can be analysed.

Target tracking is another field showing interest in Doppler radars. Battistelli et al. [47] study the tracking of a moving target by a network of Doppler sensors. Armstrong and Holeman [48] use a network of Doppler radars to determine the position, velocity and acceleration of a moving target. Cramer-Rao bound analysis was also performed, and 3-D tracking of a baseball using a network of four Doppler radars was demonstrated. Ristic and Farina [49] study joint detection and tracking of a target using multi-static Doppler-only measurements. The measurements are corrupted with additive noise as well as false detections and missed detections. A Bernoulli particle filter is used to solve this problem. Shames et al. [50] study the minimum number of Doppler-shift measurements required in a network, to generate a finite number of solutions for target position and velocity estimation. Inclusion of other measurements such as bearing and distance to the target was also studied along with the optimal placement of Doppler-shift sensors and transmitters to estimate target velocity.

Doppler radars have also frequently been used to measure speeds and velocities in sporting applications such as measuring: handball throwing speed [51], softball speed [52], velocity of a badminton shuttlecock [53], velocity of waterpolo throwing speed [54] and tennis stroke speed [55]. Doppler radars can also be used in radar guns.

The application of Doppler radars to mobile robot localisation is very different to that of target tracking in the aforementioned references. Target tracking generally describes the problem of having known, static sensors estimating the unknown state of a moving target. Conversely, the mobile robot localisation problem assumes known, static landmarks with a mobile robot carrying the sensor trying to estimate its own unknown pose.

Doppler sensors have also been employed in other navigation applications especially in maritime and aeronautical applications. The Doppler frequency shift is used to infer velocity data which can be used to complement proprioceptive data, especially to counteract non-systematic errors. For example, if the wheels are rotating, and these rotations are detected by the wheel encoders, but the robot is not moving due to obstruction by an object. Or, in the case of maritime and aeronautical applications, the Doppler frequency shift occurs when the vehicle is moving relative to the earth’s surface, therefore eliminating dead-reckoning errors induced by ocean or air currents that
may not have been accounted for \cite{56}. In a simple example, a ship’s motor may be running, implying that the boat should be moving. However, it may in fact be stationary due to a current in the opposite direction with the same force. The Doppler sensor would be able to infer that no actual motion has occurred relative to the ocean floor. This application of Doppler measurements is different to the problem of Doppler radar-based mobile robot localisation studied in this thesis. In this thesis, a mobile robot carries a Doppler radar which can be used to detect the movement of a robot relative to landmarks to help infer the robot’s absolute position in a known map, in an environment where GPS is unavailable.

Amundson et al. \cite{57} as well as Kusy et al. \cite{58}, use Doppler shift data for similar purposes - to localise a moving robot/person carrying a transmitter. One critical difference is that Amundson et al. and Kusy et al. both assume the environment contains sensors that have the ability to detect and process the transmitted signal. This requires that the environment has the necessary hardware set up in preparation for the arrival of the robot. In this thesis, the only assumption about the environment is that the map is known. There are no other hardware or equipment requirements for the environment.

There are also different environments where weaknesses can be found in different sensors. For example, LIDARs can have difficulty operating in environments that have a lot of glass or surfaces that effectively absorb light. Low-end cameras may perform very poorly in the dark. Doppler radars require the robot to move to generate a Doppler shift signal. Using multiple (types of) sensors can help to observe the environment better than using an individual sensor. For example, distance data from a sonar sensor can be highly ambiguous in a hallway with rooms, whereas there could be unique visual features that can be identified by a camera. Kortenkamp and Weymouth \cite{14} use a sonar sensor to suggest a location where an image should be taken by a camera to reduce ambiguity in a topological map. Flynn \cite{59} uses a sonar and an infrared sensor on a mobile robot.

Other sensors also have hybrid features, such as Microsoft’s Kinect which has the ability to capture RGB images like a traditional camera and also measure depth/distance, so it can be used to capture 3D data of an environment, which can also be reduced to 2D if the user wishes. The Kinect is also very modestly priced. Because of these features, the Kinect has been used in mobile robotics applications as a sensor in instances such as: \cite{60,61,62}.

For more details, DeSouza and Kak \cite{63} present a survey paper on various sensors, techniques and environment modelling for mobile robot navigation.
2.5 Algorithms for Mobile Robot Localisation

Many of the popular algorithms for mobile robot localisation are based on Bayesian filtering. The goal of Bayesian filtering is to estimate the posterior probability distribution of the robot’s pose. Bayesian filtering requires the previous posterior distribution of the robot’s pose, control input data and measurement data. Bayesian filtering consists of two steps, commonly referred to as “prediction” and “update”. Prediction involves processing the current control input signal ($u_k$) to create a probabilistic representation of the robot’s pose, following the execution of the control signal. This is modelled using the transitional/proposal density which is generally denoted as $\pi(x_k|x_{k-1})$ (the dependency on $u_k$ will be explicitly established in Chapter 2.5.2 when the filtering problem is adapted for mobile robot localisation applications). In the update step, the data from the exteroceptive sensor at the current discrete-time ($Z_k$) is then used to create a measurement probability (likelihood) distribution, which is generally denoted as $\psi(Z_k|x_k)$ which is used to update the probability distribution created by the prediction step (a dependency on the map $m$ will be explicitly established in Chapter 2.5.2 when the filtering problem is adapted for mobile robot localisation applications). The update step results in the estimated posterior distribution of the robot’s pose. Bayesian filtering therefore follows the steps:

First, using the posterior probability density function (PDF) at the previous discrete-time index, $p(x_{k-1}|Z_{1:k-1})$, apply the prediction step using the Chapman-Kolmogorov equation to obtain the prior predicted distribution:

$$p(x_k|Z_{1:k-1}) = \int \pi(x_k|x_{k-1}) \cdot p(x_{k-1}|Z_{1:k-1}) \, dx_{k-1}$$  \hspace{1cm} (2.1)$$

Then, the measurement received at discrete-time index $k$ is used to update the predicted distribution using Bayes’ rule:

$$p(x_k|Z_{1:k}) = \frac{\psi(Z_k|x_k) \cdot p(x_k|Z_{1:k-1})}{\int \psi(Z_k|x_k) \cdot p(x_k|Z_{1:k-1}) \, dx_k}$$  \hspace{1cm} (2.2)$$

resulting in the posterior distribution, $p(x_k|Z_{1:k})$. Note that the denominator of Equation 2.2 exists to ensure the posterior distribution will integrate to 1.
2.5. ALGORITHMS FOR MOBILE ROBOT LOCALISATION

The prediction step from Equation 2.1 and the update step from Equation 2.2 form the recursive process for Bayesian filtering. These concepts are further explained in Arulampalam et al. [64, p. 175], Ristic et al. [65, Chapter 1.2] and Thrun et al. [5, Chapter 2.3, Chapter 2.4].

However, implementation of the Bayes filter is dependent on various assumptions regarding the robot’s motion model, measurement model, and initial pose distribution. In some situations, these assumptions can lead to closed-form solutions, such as the famous Kalman filter for linear, Gaussian systems [66]. In a general setting, the solution can be formulated using a sample-based approximation of probability distributions, referred to as a particle filter. The choice of the implementation method also depends on factors such as the computation time, accuracy, and difficulties in implementation [5, Chapter 2.4]. Kalman filters and particle filters will now be discussed.

2.5.1 Kalman filters

Under the assumptions that the state transitional model and the measurement model are linear, and the transition and measurement noises are Gaussian, the resulting formulation of the Bayes filter is the famous Kalman filter [66]. A wealth of literature on Kalman filtering exists, for example, a tutorial by Welch and Bishop [67].

For many applications, the state transition model and/or the measurement model are not linear. The extended Kalman filter (EKF) can be used in these instances where the non-linear models are simply linearised using Taylor series approximation. The EKF is “probably the most widely used estimation algorithm for nonlinear systems” [68] and is also used in many mobile robotics applications including localisation. EKF localisation is most commonly applied when using feature-based maps [5, p. 232]. For example, Leonard, and Durrant-Whyte [69] refer to the EKF for many uses in robotics. Negenborn’s thesis [70] covers the application of Kalman filters (including EKFs) to robot localisation. Ganganath, and Leung [71] compare the EKF against a particle filter for mobile robot localisation using odometry and a Microsoft Kinect. Kong et al. [72] also localise a mobile robot with wheel encoders and a laser range finder using the EKF by extracting corners in an environment to represent features in the map. Kong et al. have succeeded in simulation and in an experiment. Similarly, Chen et al. [73] apply the EKF to odometry and laser range data for localisation. Corner angles in
the environment were also used to represent features in the map. Teslic et al. \cite{74} localise a four-wheeled robot equipped with wheel encoders and a laser range finder using an EKF. The environment can be represented by series of line segments because it is structured. The EKF is compared against more complex algorithms in a mobile robot localisation experiment by Gutmann and Fox \cite{75}. The results show that the EKF estimation accuracy is not as good as using other algorithms including particle filter-based solutions. However, the EKF computation time is also significantly faster.

The unscented Kalman filter (UKF), developed by Julier and Uhlmann \cite{68, 76}, also aims to deal with non-linear models. The UKF handles non-linearities using the unscented transformation to estimate the propagation of the mean and covariance of a probability distribution through a non-linear function. This produces better results than the linearisation technique used in the extended Kalman filter \cite{68, 76}. Further details on the UKF can be found by Thrun et al. \cite[Chapter 3.4, Chapter 7.7]{5}. In comparison, EKF linearisation is completed using Jacobians which can produce unstable results if the sampling interval is too large. Furthermore, the derivation of the Jacobian may also be non-trivial, and require a lot of computational resources especially as the sampling interval decreases \cite{77}. Ko et al. \cite{78} provide an example of the application of the UKF to mobile robot localisation. There are also a number of comparisons between the EKF and the UKF when applied to mobile robot localisation. For example, Martinelli \cite{79} performs mobile robot localisation in two scenarios. First, a laser range finder is used to measure distance to the walls in the known environment. Second, distance is measured based on RFID data in a feature-based map comprising of RFID tags as the landmarks. Giannitrapani et al. \cite{80} localise a spacecraft based on angle-only measurements. D’Alfonso et al. \cite{81} use a robot equipped with an ultrasonic sensor to achieve localisation despite using sensor switching to save battery power. The EKF and the UKF can deal with non-linear systems to a certain degree. However, for systems which are highly non-linear, their performances can be prone to divergence.

The Kalman filter assumes that the transition, and measurement noises are Gaussian distributed and that the systems are linear. Under these assumptions (along with a Gaussian initial distribution), the posterior is always a Gaussian \cite[Chapter 3.2]{5}. If the noise encountered were non-Gaussian (or especially, if it were multimodal) the performance of Kalman filters would be reduced \cite[p. 53]{82}. Furthermore, the robot could also plausibly be in a number of different locations (i.e. the posterior distribution of the robot’s
pose is multimodal) [21], especially if the environment has some symmetric properties or when solving the global localisation problem where there may be a number of high-probability hypotheses of the robot’s (initial) pose [83]. The Kalman filter’s unimodal, Gaussian distributions would not be a good representation of these scenarios. Possible Kalman filter-related solutions could include the Gaussian sum filter [84], which represents the posterior distribution as a weighted mixture of Gaussians and operates by essentially running several Kalman filters in parallel. Recently, a multimodal Kalman filter has been applied to robot localisation by Quinlan and Middleton [85] which has the capacity to maintain a multimodal posterior density. Schoenberg et al. [86], [87] apply the Gaussian sum filter with a “condensation” algorithm (to limit the number of Gaussians to manage computational resources) to achieve localisation of a vehicle equipped with odometry and multimodal vision-based measurements. Furthermore, in a scenario where the map is not known, Kwok et al. [88] solved the SLAM problem using a Gaussian sum filter consisting of parallel EKFs. Redundant EKFs were removed to save on computation costs. Kwok et al. used a robot equipped with a laser scanner, and camera but solved SLAM using bearings-only data which created additional uncertainty that could be modelled with a sum of Gaussians. The environment was represented with landmarks. Piasecki [89] combines multiple hypothesis tracking with the EKF for localisation using an ultrasonic range finder.

2.5.2 Particle filters

In light of the limitations of the Kalman-type filters, one might ask: “what kind of filters can deal with non-linear, non-Gaussian problems?”. The particle filter (PF) has become immensely popular in robotic applications [5, Chapter 4] where the posterior density is approximated by a set of weighted random samples, or “particles”. Each particle represents a hypothesis of the robot’s pose and the weight of each particle represents how likely that hypothesis is. The collection of particles and their weights form a probabilistic estimate of the robot’s pose. Particle filters can be implemented to deal with highly non-linear, non-Gaussian problems and are extremely flexible. They can also approximate a multimodal distribution. In exchange for flexibility, the particle filter is infamous for being computationally intensive [82, p. 54] unlike Kalman filters which involve closed-form matrix calculations. A number of influential tutorials on particle filters exist, such as: Arulampalam et
CHAPTER 2. BACKGROUND AND LITERATURE REVIEW

al. [64], Cappe et al. [90] and Doucet and Johansen [91]. Gustafsson [82] also wrote a tutorial on particle filters including code. Gustafsson’s tutorial discusses a number of considerations when implementing particle filters and also, describes a number of examples that are conceptually similar to the mobile robot localisation scenarios in this thesis: an underwater vessel can localise itself based on comparing sonar readings to a chart of the seabed combined with motion data such as propeller speed, rudder angle or an IMU; similarly, a plane with an IMU and a radar pointing at the ground can compare measurements against a topographic map of the land; a ship on the surface can localise itself with measurements from a radar pointing at the coast which is compared against a map of the coastline; a car can localise itself from knowing whether it is on-road or off-road compared to a roadmap along with speed and yaw rate data.

Particle filters approximate the posterior density as a weighted set of particles:

\[
p(x_k|Z_{1:k}) \approx \sum_{i=1}^{N} w_k[i] \delta(x_k - x_k[i])
\]  

where \(p(x_k|Z_{1:k})\) is the posterior density of the pose \(x_k\), given the set of measurements from discrete-time index 1 to \(k\) \((Z_{1:k})\). This posterior density is approximated by the set of particles \(x_k[i]\) and corresponding weights \(w_k[i]\), with index \(i = 1, 2, \cdots, N\). The weights should be normalised so that \(\sum_{i=1}^{N} w_k[i] = 1\) [64, p. 178] with the weight of each particle: \(0 \leq w_k[i] \leq 1\), \(\forall i\). \(\delta(\cdot)\) is the Dirac delta function [64, p. 176], [82, p. 57].

It can be shown that as the number of particles approaches infinity \((N \to \infty)\) then Equation 2.3 approaches the true posterior probability density \(p(x_k|Z_{1:k})\). This is however, at the expense of increasing the computation time.

The sequential importance sampling (SIS) particle filter consists of the following steps as explained by Ristic et al. [65, p. 39] and Arulampalam et al. [64, p. 178]:

For \(i = 1, 2, \cdots, N\):

1. Particle \(i\) is drawn from the importance/proposal distribution \(q(\cdot)\). This is the “prediction” step:

\[
x_k[i] \sim q(x_k|x_{k-1}, Z_k)
\]
2.5. ALGORITHMS FOR MOBILE ROBOT LOCALISATION

2. Particle \( i \) is assigned a weight. Using the current measurements \( (Z_k) \), the *unnormalised* weight is equal to the measurement likelihood \( \psi(Z_k|\mathbf{x}_k^{[i]}) \) multiplied by the transitional density \( \pi(\mathbf{x}_k^{[i]}|\mathbf{x}_k^{[i]-1}) \), multiplied by the previous weight \( (w_{k-1}^{[i]}) \) and divided by the proposal distribution. This is the “update” step:

\[
\tilde{w}_k^{[i]} \propto w_{k-1}^{[i]} \cdot \psi(Z_k|\mathbf{x}_k^{[i]}) \cdot \pi(\mathbf{x}_k^{[i]}|\mathbf{x}_k^{[i]-1}) \frac{q(\mathbf{x}_k^{[i]}|\mathbf{x}_k^{[i]-1}, Z_k)}{q(\mathbf{x}_k^{[i]}|\mathbf{x}_k^{[i]-1})} \quad (2.4)
\]

3. Then the weights must be normalised so that they sum to 1:

\[
w_k^{[i]} = \frac{\tilde{w}_k^{[i]}}{\sum_{p=1}^{N} \tilde{w}_k^{[p]}} \text{ for } i = 1, 2, \ldots, N
\]

After some time, using the SIS particle filter will result in particle “degeneracy” (also called depletion, or impoverishment [82, p. 58] which is the situation where many particles end up having a small/negligible weight compared to one or a few particles carrying the majority of the weight. Particle filter degeneracy is a problem because it causes computational resources to be spent on low-likelihood hypotheses (the particles with low/negligible weights) and also, a probability distribution is not represented well by only a few (or in an extreme case, one) particles with high weight. This is not a good representation of the posterior. Ideally, the posterior would be represented by many particles with similar weights.

This problem can be overcome by a process called “resampling” which is a procedure where the particles are sampled with replacement [82]. The probability of drawing the particle is proportional to its weight. Statistically, this should cause low-weight particles to not be drawn, and therefore removed from the resampled particle set. The high-weight particles should be drawn multiple times and are therefore replicated in the resampled particle set. Having more (unique) particles in the high-likelihood regions means the posterior probability distribution is more accurately represented. Computational resources are also spent inefficiently when there are many particles in the low-likelihood regions. Following the resampling step, the weights of the particles are made uniform, i.e. \( w_k^{[i]} = \frac{1}{N} \), for \( i = 1, 2, \ldots, N \).

This leads to the sequential importance resampling (SIR) particle filter which has three distinct modifications to the SIS particle filter [64, p. 181], [65, Chapter 3.5.1]:
1. The importance/proposal distribution: \(q(x_k | x_{k-1}^i, Z_{1:k})\) is chosen to be the transitional density:

\[
q(x_k | x_{k-1}^i, Z_{1:k}) = \pi(x_k | x_{k-1}^i) \tag{2.5}
\]

As a result of this choice for the proposal distribution, the particle weight from (2.4) becomes:

\[
\tilde{w}_k^i \propto w_{k-1}^i \psi(Z_k | x_k^i) \tag{2.6}
\]

2. SIR particle filters assume a resampling operation at every discrete-time \(k\). Since the weights are reinitialised uniformly following resampling, the multiplication by the previous weight \(w_{k-1}\) in Equation 2.6 becomes redundant. Therefore the weight of particle \(i\) becomes:

\[
\tilde{w}_k^i \propto \psi(Z_k | x_k^i) \tag{2.7}
\]

3. At every discrete-time index \(k\), the particles are resampled proportional to their weight, \(\tilde{w}_k^i\). Following resampling, the weights are reset as \(w_k^i = \frac{1}{N}\), for \(i = 1, 2, \ldots, N\) and which are already normalised.

While SIR particle filters assume resampling at every discrete-time \(k\), it is found that it is more advantageous to resample only when the particle weights have a high variance, which is a measure for degeneracy/impoverishment. Resampling too frequently requires computation cost and also reduces particle diversity \[92\] p. 386], [93]. This procedure is done by first calculating the effective sample size (ESS) based on the normalised weights. A common approximation for the ESS is \[93\] p. 3946], [92] pp. 386-387:

\[
\text{ESS} \approx \left( \frac{1}{N} \sum_{i=1}^{N} \left( w_k^i \right)^2 \right)^{-1} \tag{2.8}
\]

If the ESS is less than a user-defined threshold \(N_{\text{threshold}}\), resampling is applied. Following a resample, the weights are reinitialised uniformly. This process limits resampling until the variance of the particle weights is high \[5\], Chapter 4.3.4].

The technique of applying particle filters to solve the robot localisation problem is called “Monte Carlo Localisation” (MCL) \[83\], [94], [95]. MCL
has become one of the most popular algorithms used for localisation of mobile robots and this is explained in more detail by Thrun et al. [5, Chapter 8].

Applying standard MCL requires a few more subtle changes to the SIR particle filter. The proposal distribution in Equations 2.5 must be based on the previous pose as well as the control input to the robot. The proposal distribution is not dependent on the measurement. Also, the likelihood function in Equation 2.7 is influenced by the map and in the case of a Doppler radar, the control input. Therefore, MCL follows the steps:

For \( i = 1, 2, 3, \cdots, N \):

1. Particle \( i \) is drawn from the importance/proposal distribution:

\[
\mathbf{x}_k^{[i]} \sim \pi(\mathbf{x}_k | \mathbf{x}_{k-1}^{[i]}, \mathbf{u}_k)
\]

Where \( \pi(\mathbf{x}_k | \mathbf{x}_{k-1}^{[i]}, \mathbf{u}_k) \) is based on a probabilistic model of robot motion which now explicitly includes a dependency on the control input signal, \( \mathbf{u}_k \).

2. Particle \( i \) is assigned a weight:

\[
\tilde{w}_k^{[i]} = \psi(Z_k | \mathbf{x}_k^{[i]}, \mathbf{m})
\]

Where the likelihood now explicitly includes a dependency on the static map of the environment, \( \mathbf{m} \).

The resampling step is performed at every discrete-time \( k \). The MCL algorithm is explained in more detail in Chapter 3.6 of this thesis.

MCL and derivative works have been used in many robot localisation applications for example: Rowekamper et al. [96] use a robot equipped with a laser rangefinder for MCL. Rowekamper et al. are able to verify their pose estimate using a high precision motion capture system. Vale et al. [97] use an underwater ROV equipped with a multi-beam sonar and an approximately known map. The parallel calculations of a particle filter has been completed on a GPU to reduce computation time.

One of the advantages of applying particle filters to solve the mobile robot localisation problem is the ability to maintain a multimodal posterior distribution. This is particularly helpful in environments which have some areas of symmetry or ambiguity. Even so, it is still possible over time that the particles converge to the wrong hypothesis. This is known as the “premature
convergence” problem. To reduce the occurrence of this kind of problem, it can be advantageous to maintain a higher diversity of particles. Milstein et al. [98] apply clustered particle filtering to solve the robot localisation problem (“Cluster-MCL”) which maintain clusters of particles for hypotheses which are very distinct. Research into Cluster-MCL was also completed by Liu et al. [99]. Kootstra and Boer propose the combination of “niching” methods (which are well-known in the field of genetic algorithms), with particle filters to maintain greater particle diversity for MCL in symmetric environments [100].

Lenser and Veloso [30] created the Sensor Resetting Localisation (SRL) algorithm which is an extension of MCL. SRL was created for use in the RoboCup robotic soccer competition where robots can face unmodelled collisions and the referee can pick up the robot and move it without the robot knowing (i.e. kidnapping and localisation failure). The measurement and odometry readings are also subject to errors and noise. These factors adversely affect the results of standard MCL. SRL overcomes these problems by drawing a number of particles from the sensor’s probability distribution. If the probability of the current localisation estimate being correct is poor i.e. the robot is probably not where it thinks it is, then a greater number of particles are drawn from the alternative probability distribution based on the current measurement. However, the ability to draw particles from the sensor’s probability distribution is only possible in certain conditions [5, Chapter 8.3.6]. Thrun et al. [101] argue that the algorithm from Lenser, and Veloso does not approximate the posterior, so they extend SRL into “mixture-MCL”. Mixture-MCL can also be traced to the work published a year earlier by Thrun et al. [102].

The minimum mean square error (MMSE) estimator is used as the final estimate of the robot’s pose. Using the posterior approximated with particles, the MMSE is defined as follows [103], [104]:

\[
\hat{x}_{k}^{MMSE} = E[x_k | Z_{1:k}] \triangleq \int_{-\infty}^{\infty} x_k \cdot p(x_k | Z_{1:k}) \, dx_k \approx \sum_{i=1}^{N} w_{k}^{[i]} x_{k}^{[i]}
\]

In the three dimensions of the pose, this leads to:

\[
\hat{x}_k = \sum_{i=1}^{N} w_{k}^{[i]} x_{k}^{[i]}
\]  

(2.9)
\[ \hat{y}_k = \sum_{i=1}^{N} w_k^{[i]} y_k^{[i]} \] (2.10)

\[ \hat{\theta}_k = \sum_{i=1}^{N} w_k^{[i]} \theta_k^{[i]} \] (2.11)

The MMSE will be used throughout this thesis.

**Particle Filters and Proposal Distributions**

One key challenge for particle filters is finding an *importance / proposal* density that closely approximates the posterior distribution. For typical robotics applications, the importance density is based purely on a probabilistic model of the robot’s movement actuated by control input signals. The control input signal is typically either: input signals to the motors, wheel rotation data or IMU data. This proposal distribution can be relatively diffuse and inaccurate compared to the data returned from the robot’s exteroceptive sensors, such as LIDAR (light detection and ranging) data [105]. As a consequence, this sometimes causes the problem of weight *degeneracy*: after the data from the sensor is applied to update the predicted density, only a few particles have a significant weight and the rest of the particle sets have negligible, or zero, weight [93]. Particle weight degeneracy is a problem because it means that the posterior distribution is represented by a few (or in an extreme case, one) highly-weighted particles. This is not a good representation of the posterior. Ideally, the posterior would be represented by many particles with similar weights. In this situation where the proposal distribution is not well-selected, the particle filter requires a larger amount of particles to sufficiently cover the region with a high likelihood [105], therefore increasing computation costs. These problems could be mitigated by selecting a proposal distribution that better resembles the posterior distribution.

To illustrate another consequence of a poorly selected proposal distribution, consider an extreme example of when the proposal distribution is a complete mismatch of the posterior: all of the particles are first drawn from the proposal distribution. If these particles are too far away from the true posterior distribution then the particle filter will fail to converge to the true pose. Therefore, having more particles drawn from a proposal distribution that is similar to the posterior distribution means that the true posterior distribution will be estimated with greater accuracy.
State-of-the-art research attempts to incorporate data from the sensor to influence the proposal distribution. With the proposal distribution incorporating data from the sensor, the particles should have more resemblance to the posterior. Cappe et al. [90, Section II D] refer to the optimal kernel as incorporating information from the state dynamic as well as the current observation. However, this optimal kernel is intractable most of the time, so a large part of research work attempts to approximate this kernel. Quang et al. [106] also explain this idea clearly. A number of techniques attempt to incorporate the latest measurement into forming the proposal distribution. Quang et al. [106] have developed the Laplace particle filter which uses the Laplace approximation to incorporate the latest measurement into the proposal distribution. The “Gmapping” proposal distribution by Grisetti et al. [105] is a very influential proposal distribution in robotics. It is an extension of their work completed in 2005 [107] which was itself, an improvement on the algorithm described by Hahnel et al. [108]. Importantly, it requires a numerical optimisation step where each particle is moved around the map to find the pose that maximises the likelihood of the received LIDAR measurement. This will be explained in more detail in Chapter 5.2.

Other methods of improving the proposal distribution also exist. Beeson et al. [109] suggest two methods of improving the proposal distribution: First, Beeson et al. suggest shrinking the proposal distribution. Since the likelihood is relatively narrow compared to the motion model proposal distribution, the likelihood is approximated by a Gaussian which becomes the new proposal distribution. This makes the prior distribution a more accurate representation of the posterior and hence, a higher accuracy can be achieved with less particles. This incorporated methods from [107], but appears to be a simpler implementation. Second, Beeson et al. suggest growing the proposal distribution. This is used to handle the scenario where the proposal distribution has a small, or no intersection with the high-likelihood regions. Consider an example where the robot’s wheels are turning, but the robot itself is not moving as it has been obstructed by an obstacle. If the proposal distribution were based purely on the wheel encoding data, the proposal distribution might assume the robot has been moving far away. This situation is detected when the likelihood of the particles fall below a threshold, then the proposal distribution is grown larger.

Also discussed in this thesis is the Exact Daum-Huang (EDH) particle flow particle filter (PFPF) [110, 112] explained in Chapter 3.7. Under some limiting assumptions, the particles incorporate the measurement data and
“flow” to the posterior distribution. This is beneficial as it avoids the computationally intensive step of scan-matching or numerically finding the maximum likelihood pose as per the Gmapping proposal distribution. However, the EDH PFPP requires the calculation of a Jacobian matrix.

The unscented particle filter (UPF) [113] uses an unscented Kalman filter (UKF) as the proposal distribution for the particles. Doing so also incorporates the latest available information when drawing particles from the proposal distribution which draws them in regions of high likelihood. As discussed earlier, this is very useful if the likelihood is relatively narrow compared to the prior distribution, or if the likelihood overlaps in a particularly unlikely part of the prior.

Kim et al. [114] also apply the techniques of the unscented particle filter to the SLAM problem and have called it unscented FastSLAM (UFastSLAM). Wang and Zhang [115] use the UPF to estimate the robot pose and UKFs to estimate the landmark positions to solve the SLAM problem.

Havangi et al. [116] use particle swarm optimisation (PSO) to optimise the proposal distribution in their algorithm called the “square root unscented FastSLAM” (SRUFastSLAM). PSO causes the particles to be sampled from the high probability region of the posterior. The authors of this paper claim to have achieved better estimation accuracy than UFastSLAM.

Also worth mentioning is the existence of algorithms which do not rely on recursive Bayesian estimation techniques. Cohen and Koss [117] present a number of solutions to the mobile robot localisation problem given three landmarks and bearings-only measurements, based on triangulation and geometric arguments. One of the algorithms presented, the Newton-Raphson iterative method may be able to incorporate data from dead-reckoning sensors. Esteves et al. [118] remove some limitations of the Geometric Triangulation algorithm mentioned by Cohen and Koss [117] to produce the Generalised Geometric Triangulation algorithm. Sutherland and Thompson [119] propose a method of bearings-only localisation based on a number of carefully chosen landmarks to reduce localisation error. Madsen et al. [120] also consider bearings-only localisation using triangulation with three landmarks and study the effects of noise on the pose estimate as well as selecting three optimal landmarks from the environment to minimise pose uncertainty. Madsen et al. also make a reference to incorporating odometry data in their work [120, p. 7]. Betke, and Gurvits [121] use techniques based on geometry for mobile robot localisation on a feature-based map. The features are represented using complex numbers. This technique does not require a proposal
distribution or knowledge of the robot’s motion (such as velocity control input, odometry, IMU data etc.) and has computation time increasing only linearly with the number of landmarks. This approach relies on minimising squared error without considering sensor/motion models.

A number of limitations exist for this class of algorithms. Generally, at least three landmarks are required for successful localisation based on triangulation methods [117, p. 96], [118, p. 346], [120]. Limitations may exist on the labelling and ordering of landmarks, the geometry between landmarks, and the position of the robot relative to the landmarks [117], [118]. Some of these limitations motivated the work by Esteves et al. [118]. However, it is also worth mentioning that the number of landmarks, landmark-measurement association and geometry of the environment may also influence the results of Bayesian solutions. Chapter 3 investigates how the number of landmarks affects the amount of information available by analysing the Cramer-Rao lower bound (CRLB). Also, as previously mentioned in this chapter, localisation in symmetric/ambiguous environments can also be problematic.

In comparison to Bayesian solutions for mobile robot localisation, a number of very general observations can be made on the referenced papers. First, the triangulation/geometric solutions seem to be relatively focussed on feature-based maps, whereas Bayesian solutions may have wider applicability to other maps such as occupancy grid-based maps or topological maps. Second, the referenced triangulation/geometric solutions have more emphasis on bearings-only measurements. The positive aspect of this is that the engineering problem can potentially be solved with cheaper and lighter sensors that consume less power. The disadvantage of this is that extremely valuable and rich data such as distance, camera images etc. is not being utilised to improve the estimate of the robot’s pose, unlike Bayesian filtering solutions which frequently incorporate this data. Third, the referenced triangulation/geometric solutions only make minor references to utilising proprioceptive data from velocity control input/odometry/IMU sensors etc. As argued previously, this could result in cost/weight/power savings, however, as before, it is again, neglecting a useful source of information frequently incorporated by Bayesian solutions. Consider an example where the proprioceptive data reports that little/no motion has occurred for the robot. On the contrary, the exteroceptive sensor may report that the robot has moved (possibly due to noise in the exteroceptive sensor measurement), perhaps by a large amount. Which sensor should be trusted? The triangulation/geometric
solutions referenced generally rely on only the exteroceptive data regardless of what the proprioceptive sensor is reporting. This could be beneficial in exceptional circumstances such as if the exteroceptive sensor was so much more accurate than the proprioceptive sensor that the proprioceptive sensor was practically redundant. Bayesian filtering typically utilises a combination of proprioceptive/exteroceptive data sources. Fourth, Bayesian filtering utilises the previous pose of the robot as a source of information. Relying purely on the current measurement data to localise without consideration of the most recent pose of the robot could lead to an estimate that is practically impossible considering the robot motion dynamics. On the contrary, it could also be useful to localise based only upon the recent measurement in exceptional circumstances. Consider a scenario where a Bayesian filter fails and loses track of the robot’s pose. An algorithm that localises based only on the most recent measurement can be a means of restarting/recovering the filter as it does not consider the previous pose. Examples based on this concept were mentioned earlier in Chapter 2.5.2 see: SRL [30] and mixture-MCL [101], [102]. Fifth, Bayesian filtering incorporates a probabilistic model of the robot’s sensors and motion. This allows for the explicit incorporation of useful information concerning physical characteristics of the robot and its sensors and can allow for the use of realistic motion and sensor models as well as realistic noise models which can be complicated. For example, Chapter 4 of this thesis assumes the sensor can produce false detections as well as miss a landmark. It is also assumed that the landmark-measurement associations are unknown. In comparison, it is unclear on how well triangulation/geometric methods can incorporate more complicated assumptions on system/noise modelling. Finally, some Bayesian filters such as particle filters can maintain multiple hypotheses which can be useful in situations where there is currently insufficient information to reach a conclusion, ambiguous/symmetric environments or high measurement noise.

Despite the proposed benefits of Bayesian filters, the approaches based on geometric/triangulation techniques are a different, and also, intelligent way of approaching the same localisation problem. Bayesian filters could likely be improved by incorporating many of the arguments from geometric/triangulation-based reasoning as it could be considered as another source of data and constraints, that could be utilised to improve estimation accuracy.
2.6 Computation

Particle filters are exceptionally flexible in dealing with complicated problems where the models can be non-linear and the noise can be non-Gaussian. In exchange for this flexibility, the major shortcoming of particle filters is the computation time required to solve the problem. The primary consequence of this is not being able to use the particle filter in high-speed applications.

For the mobile robot localisation problem, the uncertainty is the greatest initially, when the initial robot pose is unclear, or, in some problems, no knowledge of the initial pose is assumed to be available like in Chapter 5.5 and in Chapter 4 of this thesis. These situations require a large number of particles to cover the state space.

However, as more proprioceptive and measurement data is received, the particle filter becomes more and more certain about the robot’s pose and the filter is supposed to converge. Therefore, a smaller number of particles is required to maintain an accurate estimation. This is the principle behind KLD-sampling [5], [122], explained in Chapter 4.4.1 of this thesis. Soto [123] claims that the number of particles calculated using KLD-sampling is based on the assumption that the empirical proposal distribution comes from the true distribution and has developed an alternative calculation for the number of particles without this assumption.

Running the particle filter requires similar computations to be carried out on many different particles, meaning that this class of filters can potentially benefit from parallelisation. The simplest implementation is to divide the particles into a number of subsets and have each subset run the motion model and weight calculation on a different processor on a multicore processor [124, p. 1192]. Newer hardware solutions are possible, including general-purpose graphical processing units (GPGPUs), field-programmable gate arrays (FPGAs) and computer clusters on a network [125]. GPGPUs, or simply referred to as GPUs, have been gaining interest in fields such as artificial intelligence and recently, have been applied to particle filters. GPUs are most commonly known for their use in producing graphics for computers to be displayed on a screen. GPUs are optimised to work with large amounts of data in parallel. Hence, algorithms which can be parallelised can potentially benefit by being executed on a GPU rather than a central processing unit (CPU).

The work by Hendeby et al. [125], [126] proposed the first general implementation of the particle filter on a GPU. The results show that for a low number of particles, the overhead of initialising and utilising the GPU
put it at a disadvantage compared to a CPU implementation. However, as the number of particles becomes very large, the GPU has a slight advantage. The implementation on the GPU however, still has some reliance on the CPU (mainly, the random number generator) which has slowed down the execution speed. Further information can also be found in Hendeby’s thesis [127]. In addition to (sometimes prior to) the work by Hendeby et al., numerous other implementations of the particle filter involving a GPU for specific applications were completed in: visual tracking [128], [129], bearings-only tracking [130] and sonar Monte Carlo Localisation of an underwater remotely operated vehicle (ROV) [97].

Particle filters were implemented using an FPGA in a number of other applications, such as: radar tracking [131] and object tracking [132].

Some operations such as applying the motion model or calculating the weight of a particle, are independent and can be readily parallelised. However, the resampling operation is collective and usually requires waiting for the completion of the calculation of all of the particle weights and is considered to be a potential bottleneck for large-scale problems. This makes the parallelisation of alternative resampling algorithms an area of research [133]–[136]. Schwiegelshohn et al. [137] studied parallel resampling algorithms implemented on an FPGA.

Liu et al. [138] claim that the majority of computation time on GPU implementations of the particle filter is spent on resampling, especially when the number of particles is large. Liu et al. present various resampling algorithms on an FPGA and show that the FPGA computation speeds for resampling are faster than the GPU implementations.

Chitchian et al. [139] propose a particle filter consisting of a number of local sub-filters that can exchange information with each other. This filter can take advantage of the distributed computation advantages of a GPU and can be used in real-time applications.

A secondary consequence of slow computations is the loss of data which can lead to sub-optimal estimation. With the improvement in modern sensor capabilities, new sensor readings often arrive before the computer can complete the update of the particle filter. The simplest way to handle the excess sensor readings is to discard it. However, Kwok et al. [140], [141] have developed a way to incorporate the large amounts of sensor data arriving in real-time across different sets of particles. The posterior is then a weighted mixture of these sets of particles. Their algorithm is called the real-time particle filter (RTPF) [141], or when KLD-sampling is incorporated into the
algorithm, it is called the adaptive real-time particle filter (ARTPF) \[140\].

\section*{2.7 Conclusion}

This chapter covered a number of areas of research for mobile robot localisation. Localisation is performed in different environments with maps commonly represented using: feature-based maps, occupancy grid maps, topological maps and modern combination maps.

There are various assumptions that can make the localisation problem more difficult. For example, is there any information about the initial pose of the robot? What if localisation fails/the robot gets “kidnapped”? What if the environment is full of people/the environment is changing?

A number of sensors were also reviewed. Ultrasonic sensors were previously popular but have largely been replaced by cheap, 2-D LIDARs. Visual sensors such as cameras have recently been highly researched \[2\]. The potential application of Doppler radars for robot localisation is a key focus of this thesis. Modern hybrid sensors such as the Microsoft Kinect (which is able to capture RGB images along with distance) have also been used in mobile robot localisation.

How is the localisation problem solved? Bayesian filtering schemes are amongst the most popular choice to solve the mobile robot localisation problem in modern times. Extended Kalman filters, and unscented Kalman filters are used in many of the non-linear, localisation applications, in addition to their widespread usage in the field of engineering in general. However, when dealing with (highly) non-linear, non-Gaussian problems, one may consider using particle filters. The application of particle filters to mobile robot localisation is called “Monte Carlo Localisation” (MCL) \[83, 94, 95\]. MCL has become one of the most popular algorithms used for localisation of mobile robots \[5, Chapter 8\].

Computation time is also an important consideration for real-time robotics, especially for particle filters which are flexible, but computationally expensive. Research in reducing computation time focuses on improvements to the algorithms, and also, parallelised hardware solutions such as GPUs and FPGAs.
Chapter 3

Localisation using Doppler Radar

3.1 Introduction

This chapter investigates the use of Doppler-azimuth radars in a feature-based map for mobile robot localisation in a GPS-denied environment, under some basic assumptions. Doppler radars are studied because they are potentially cheaper, lighter, and consume less power than some other robotics sensors. However, they are also less informative than LIDAR as they return measurements based on the motion of the robot relative to the fixed landmark rather than the range and azimuth to the landmark as reported by a LIDAR. The application of Doppler-azimuth radars to solve the mobile robot localisation problem was studied in simulation on feature-based maps. The Doppler-azimuth radar transmits a wave and the movement of the mobile robot relative to the static, known landmarks (features) in the map causes a detectable Doppler frequency shift in the returning wave. Based on these measurements, the pose of the mobile robot is to be estimated.

This chapter will study the feasibility of using a Doppler radar for mobile robot localisation in a GPS-denied environment by analysing the Cramer-Rao lower bound (CRLB) which will indicate in theory, the most accurate estimation possible for this scenario. The CRLB is also compared against the achieved root mean square error (RMSE) of the EKF. Scenarios were also considered where the measurement noise for the Doppler signal was extremely high (meaning, that the measurements are akin to being azimuth-
only) and conversely, when the measurement noise for azimuth was extremely high (meaning, that the measurements are akin to being Doppler-only).

In this work, it is assumed that there is an array of Doppler radars spanning $2\pi$ radians field of view fixed to a mobile robot. This array is able to provide both Doppler and azimuth measurements for robot localisation. The radar has the additional characteristics: All landmarks are hit by the radar i.e. landmarks cannot be missed by the radar; the radar does not produce any false detections; all landmark-measurement associations are known; and the measurement noise is Gaussian.

The initial distribution of the robot’s pose is also assumed to be known.

Three filtering algorithms are studied in this chapter: the extended Kalman filter (EKF), the sequential importance resampling (SIR) particle filter - A standard particle filter, introduced in Chapter 2.5.2 and the Exact Daum-Huang (EDH) particle flow particle filter - A state of the art particle filter where the particles flow towards the posterior distribution based on the likelihood and measurement.

The results of Chapter 3.5 were published by Guan et al. to a journal paper [6] and a conference paper [9].

3.2 Motion Model

The robot’s pose at discrete-time index $k$, is a vector $\mathbf{x}_k = [x_k, y_k, \theta_k]^T$. $(x_k, y_k)$ are Cartesian coordinates of the robot’s location and $\theta_k$ is its heading. The robot’s motion is actuated by a control input applied from $k - 1$ to $k$. The motion model is of the form:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{e}_k \tag{3.1}$$

The noise $\mathbf{e}_k$ can occur due to various sources such as: obstacles, uneven terrain or slippage between the robot’s wheels and the floor which prevent the control input signal from actuating the robot as intended. Otherwise, noise is also inherent in the proprioceptive sensors, such as IMUs.

3.2.1 Velocity Motion Model

Referring to Equation (3.1), the velocity motion model is described by Thrun et al. [5] Chapter 7.4.3, Chapter 5.3.3] where the control input applied from $k - 1$ to $k$ comprises of a translational and a rotational velocity, denoted by:
3.2. MOTION MODEL

\( u_k = [v_k, \omega_k]^T \). The function \( f(\cdot) \) from Equation 3.1 is a \( 3 \times 1 \) vector with the following components that describe the robot’s motion:

\[
f_1(x_{k-1}, u_k) = x_{k-1} - \frac{v_k}{\omega_k} [\sin \theta_{k-1} - \sin(\theta_{k-1} + \omega_k \Delta t)]
\]

\[
f_2(x_{k-1}, u_k) = y_{k-1} + \frac{v_k}{\omega_k} [\cos \theta_{k-1} - \cos(\theta_{k-1} + \omega_k \Delta t)]
\]

\[
f_3(x_{k-1}, u_k) = \theta_{k-1} + \omega_k \Delta t
\]

\( \Delta t = t_k - t_{k-1} \) is the sampling interval.

Referring to [5, pp. 205-206], the motion noise is originally assumed to be added to the control input \( u_k \). However, in order to be compatible with the extended Kalman filter state space assumptions, the motion noise is assumed to be distributed according to a zero-mean, multivariate Gaussian distribution: \( e_k \sim \mathcal{N}(0_{3 \times 1}, Q_k) \) and is added to the motion model \( f(\cdot) \) in the state space, as per Equation 3.1. This therefore requires that the motion noise in the control input space be converted to motion noise in the state space. Let the diagonal matrix:

\[
D_k = \begin{bmatrix}
\gamma_1 v_k^2 + \gamma_2 \omega_k^2 & 0 \\
0 & \gamma_3 v_k^2 + \gamma_4 \omega_k^2
\end{bmatrix}
\]

be the covariance of the noise for the control inputs. \( D_k \) has parameters \( \gamma_1, \gamma_2, \gamma_3, \) and \( \gamma_4 \) which should reflect noise levels.

The approximation of the covariance of the motion noise in state space, is:

\[
Q_k \approx B_k D_k B_k^T
\]

which is an approximate mapping of motion noise from the control input space, to the state space.

The matrix \( B_k \) is a Jacobian used to approximately transform the noise from the control input space to the state space. \( B_k \) is defined as:

\[
B_k = \frac{\partial f(x_{k-1}, u_k)}{\partial u_k} \bigg|_{x_{k-1}, u_k} = \begin{bmatrix}
\frac{\partial f_1}{\partial v_k} & \frac{\partial f_1}{\partial \omega_k} \\
\frac{\partial f_2}{\partial v_k} & \frac{\partial f_2}{\partial \omega_k} \\
\frac{\partial f_3}{\partial v_k} & \frac{\partial f_3}{\partial \omega_k}
\end{bmatrix}
\]

\( x_{k-1}, u_k \)
where:

\[
\begin{align*}
\frac{\partial f_1}{\partial v_k} &= \frac{1}{\omega_k} \{ \sin(\theta_{k-1} + \omega_k \Delta t) - \sin \theta_{k-1} \} \\
\frac{\partial f_1}{\partial \omega_k} &= \frac{v_k}{\omega_k^2} (\sin \theta_{k-1} - \sin(\theta_{k-1} + \omega_k \Delta t)) \\
&\quad + \omega_k \Delta t \cos(\theta_{k-1} + \omega_k \Delta t) \\
\frac{\partial f_2}{\partial v_k} &= \frac{1}{\omega_k} \{ \cos \theta_{k-1} - \cos(\theta_{k-1} + \omega_k \Delta t) \} \\
\frac{\partial f_2}{\partial \omega_k} &= \frac{v_k}{\omega_k^2} (-\cos \theta_{k-1} + \cos(\theta_{k-1} + \omega_k \Delta t)) \\
&\quad + \omega_k \Delta t \sin(\theta_{k-1} + \omega_k \Delta t) \\
\frac{\partial f_3}{\partial v_k} &= 0 \\
\frac{\partial f_3}{\partial \omega_k} &= \Delta t
\end{align*}
\]

The transitional density can therefore be expressed as:

\[
\pi(x_k|x_{k-1}, u_k) = \mathcal{N}(x_k; f(x_{k-1}, u_k), Q_k)
\] (3.8)

### 3.3 Measurement Model

At discrete-time index \( k \), the received measurement \( Z_k \) is modelled by:

\[
Z_k = h(x_k, m, u_k) + n_k
\] (3.9)

where \( m \) is the map of the environment. The measurement noise is assumed to be Gaussian distributed with zero mean and covariance matrix of \( R_k \):

\[
n_k \sim \mathcal{N}(0, R_k)
\]

The measurement equation \( h(x_k, m, u_k) \) is defined below for the Doppler-azimuth radar:

\[
h(x_k, m, u_k) = \left[ d_k^{(1)}, \phi_k^{(1)}, d_k^{(2)}, \phi_k^{(2)}, \ldots, d_k^{(L_k)}, \phi_k^{(L_k)} \right]^T
\] (3.10)
Note that the Doppler-azimuth measurements are a function of $u_k$ because unlike a LIDAR, the measurement from a Doppler radar is explicitly affected by the movement of the robot. The measurement set contains the Doppler frequency shift ($d_k^{(j)}$) and the azimuth measurements ($\phi_k^{(j)}$) for $j = 1, 2, \cdots, L_k$. In this scenario, the number of received measurements ($L_k$) is equal to the number of landmarks ($M$), for all $k$, because it is assumed that landmarks cannot be missed by the radar and the radar does not produce false detections. Therefore the dimensions of the covariance matrix $R_k$ is $2L_k \times 2L_k$, with $R_k$ equal to:

$$R_k = \text{diagonal} \left( \left[ \sigma_d^2, \sigma_\phi^2, \sigma_d^2, \sigma_\phi^2, \cdots, \sigma_d^2, \sigma_\phi^2 \right] \right)$$ (3.11)

For an arbitrary landmark in the map $(m^{(j)})$ located at the known coordinates $(x^{(j)}, y^{(j)})$, that has been successfully detected (hit), the Doppler shift can be expressed as:

$$d_k^{(j)} = -\frac{2f_c v_k \left\{ (x_k - x^{(j)}) \cos \theta_k + (y_k - y^{(j)}) \sin \theta_k \right\}}{c \sqrt{(x_k - x^{(j)})^2 + (y_k - y^{(j)})^2}}$$ (3.12)

and the azimuth can be expressed as:

$$\phi_k^{(j)} = \arctan \frac{y^{(j)} - y_k}{x^{(j)} - x_k} - \theta_k$$ (3.13)

$f_c$ is the carrier frequency of the radar. For the entire thesis, $f_c$ is assumed to be a constant equal to $10.525 \times 10^9$ Hz. The constant $c$ is the speed of light.

The likelihood function of a measurement during discrete-time $k$, is therefore:

$$\psi(Z_k|x_k, m, u_k) = \mathcal{N}(Z_k; h(x_k, m, u_k), R_k)$$ (3.14)

Figure 3.1 shows an approximate illustration of the scenario. The environment is modelled by a feature-based map. The mobile robot is carrying a Doppler radar emitting a signal which reflects off of landmarks in the environment producing a noisy Doppler-azimuth measurement.
CHAPTER 3. LOCALISATION USING DOPPLER RADAR

Next movement initiated by: $[v_{k+1}, w_{k+1}]^T$

Tangent to the path travelled

Landmark $j$: $m^{(j)}$

Figure 3.1: Illustration of mobile robot localisation with Doppler radar.
3.4 Cramer-Rao Lower Bound (CRLB)

The CRLB is a theoretical, best achievable second-order error performance for nonlinear filtering independent of measurements. Ristic et al. [65, Equation 4.47] explains that the CRLB is the inverse of the information matrix \( \mathbf{J}_k \) which can be calculated recursively for a linear, Gaussian scenario:

\[
\mathbf{J}_k = (\mathbf{Q}_k + \mathbf{F}_k \mathbf{J}_{k-1}^{-1} \mathbf{F}_T^T)^{-1} + \mathbf{H}_T^T \mathbf{R}_k^{-1} \mathbf{H}_k
\] (3.15)

Due to process noise, the robot’s trajectory is random, so the information matrix is conditional upon the particular robot pose trajectory.

When examining Ristic et al. [65, Equation 4.47] in comparison with Equation 3.15 there may be slight confusion on whether the matrices are evaluated at discrete-time index \( k \) or \( k + 1 \). This is because Ristic et al. [65, Equation 4.30-4.31] define \( \mathbf{Q}_{k-1} \) and \( \mathbf{R}_k \) as the covariance matrix for the current motion and current measurement, respectively, at discrete-time \( k \). However, Thrun et al. [5, Equation 7.6 and Equation 7.12] and this thesis define \( \mathbf{Q}_k \) and \( \mathbf{R}_k \) as the covariance matrix for the current motion and current measurement, respectively, at discrete-time \( k \). Upon closer inspection, there are no logical inconsistencies.

Initially, \( \mathbf{J}_0 \) is the inverse of the initial uncertainty. That is,

\[
\mathbf{J}_0 = \mathbf{P}_{0|0}^{-1}
\] (3.16)

Since the models for this application are nonlinear, the Jacobian of the nonlinear function is used. The matrices \( \mathbf{F}_k \) and \( \mathbf{H}_k \) from Equation 3.15 are Jacobians, evaluated at the true state. \( \mathbf{F}_k \) is the Jacobian of the motion model with respect to the state, to be evaluated at the true state and true control input for the purposes of calculating the CRLB. The motion model
\( f(x_{k-1}, u_k) \) can be found in Chapter 3.2.1. The Jacobian \( F_k \) is given by:

\[
F_k = \frac{\partial f(x_{k-1}, u_k)}{\partial x_{k-1}} \bigg|_{x_{k-1}, u_k}
\]

\[
= \begin{bmatrix}
\frac{\partial f_1}{\partial x_{k-1}} & \frac{\partial f_1}{\partial y_{k-1}} & \frac{\partial f_1}{\partial \theta_{k-1}} \\
\frac{\partial f_2}{\partial x_{k-1}} & \frac{\partial f_2}{\partial y_{k-1}} & \frac{\partial f_2}{\partial \theta_{k-1}} \\
\frac{\partial f_3}{\partial x_{k-1}} & \frac{\partial f_3}{\partial y_{k-1}} & \frac{\partial f_3}{\partial \theta_{k-1}}
\end{bmatrix}
\bigg|_{x_{k-1}, u_k}
\]

\[
= \begin{bmatrix}
1 & 0 & \frac{v_k}{\omega_k} \{\cos(\theta_{k-1} + \omega_k \Delta t) - \cos \theta_{k-1}\} \\
0 & 1 & \frac{v_k}{\omega_k} \{\sin(\theta_{k-1} + \omega_k \Delta t) - \sin \theta_{k-1}\} \\
0 & 0 & 1
\end{bmatrix}
\]

\( H_k \) is the Jacobian of the Doppler radar measurement model with respect to the state, to be evaluated at the true state and the true control input for the purpose of calculating the CRLB. The measurement function \( h(x_k, m, u_k) \), can be found in Equation 3.10. The Jacobian \( H_k \) is given by:

\[
H_k = \frac{\partial h(x_k, m, u_k)}{\partial x_k} \bigg|_{x_k, u_k}
\]

\[
= \begin{bmatrix}
\frac{\partial d^{(1)}_k}{\partial x_k} & \frac{\partial d^{(1)}_k}{\partial y_k} & \frac{\partial d^{(1)}_k}{\partial \theta_k} \\
\frac{\partial d^{(2)}_k}{\partial x_k} & \frac{\partial d^{(2)}_k}{\partial y_k} & \frac{\partial d^{(2)}_k}{\partial \theta_k} \\
\frac{\partial d^{(L_k)}_k}{\partial x_k} & \frac{\partial d^{(L_k)}_k}{\partial y_k} & \frac{\partial d^{(L_k)}_k}{\partial \theta_k}
\end{bmatrix}
\bigg|_{x_k, u_k}
\]

where the components are:

\[
\frac{\partial d^{(i)}_k}{\partial x_k} \bigg|_{x_k, u_k} = \frac{2f_c}{c} \begin{bmatrix}
-\frac{v_k \cos \theta_k}{\sqrt{(x_k-x^{(i)})^2+(y_k-y^{(i)})^2}} + \frac{v_k (x_k-x^{(i)})^2 \cos \theta_k}{(\sqrt{(x_k-x^{(i)})^2+(y_k-y^{(i)})^2})^3} + \frac{v_k (y_k-y^{(i)}) (x_k-x^{(i)}) \sin \theta_k}{(\sqrt{(x_k-x^{(i)})^2+(y_k-y^{(i)})^2})^3}
\end{bmatrix}
\]

\[
\frac{\partial d^{(i)}_k}{\partial y_k} \bigg|_{x_k, u_k} = \frac{2f_c}{c} \begin{bmatrix}
-\frac{v_k \sin \theta_k}{\sqrt{(x_k-x^{(i)})^2+(y_k-y^{(i)})^2}} + \frac{v_k (x_k-x^{(i)}) (y_k-y^{(i)}) \cos \theta_k}{(\sqrt{(x_k-x^{(i)})^2+(y_k-y^{(i)})^2})^3} + \frac{v_k (y_k-y^{(i)})^2 \sin \theta_k}{(\sqrt{(x_k-x^{(i)})^2+(y_k-y^{(i)})^2})^3}
\end{bmatrix}
\]

\[
\frac{\partial d^{(i)}_k}{\partial \theta_k} \bigg|_{x_k, u_k} = \frac{2f_c}{c} \begin{bmatrix}
v_k \{ (x_k-x^{(i)}) \sin \theta_k - (y_k-y^{(i)}) \cos \theta_k \} \sqrt{(x_k-x^{(i)})^2+(y_k-y^{(i)})^2}
\end{bmatrix}
\]
3.5. EXTENDED KALMAN FILTER (EKF)

The CRLB that is studied in this chapter was introduced in [142, p. 6] and is referred to as the expected conditional CRLB. It is defined as:

\[ P^*_k = E[J_k^{-1}] \]

where the information matrix \( J_k \) is given by Equation 3.15. The matrix \( P^*_k \) is computed by taking the average over random realisations of the robot pose.

The CRLB for the RMSEs of the three estimated states are then obtained through the diagonal elements of \( P^*_k \) as [65, Chapter 4, p. 91, p. 129]:

\[
\text{CRLB(RMSE)}_x = \sqrt{P^*_k(1,1)} \\
\text{CRLB(RMSE)}_y = \sqrt{P^*_k(2,2)} \\
\text{CRLB(RMSE)}_\theta = \sqrt{P^*_k(3,3)}
\]

The RMSE is one way to measure the accuracy of the state estimate. This is defined ahead in Equation 3.21.

The CRLB was calculated in order to investigate in theory, how well the robot pose can be estimated using Doppler-azimuth measurements.

3.5 Extended Kalman Filter (EKF)

The EKF is a recursive algorithm applied to estimate the state. The EKF algorithm was introduced in Chapter 2.5.1 with further explanations from Thrun et al. [5] and Bar-Shalom et al. [104].

Assuming that the initial state vector \( \hat{x}_{0|0} \), and its covariance \( P_{0|0} \), are known, the computational steps of the EKF are summarised by Guan et al. [6] below:

\[
\frac{\partial \phi^{(i)}_k}{\partial x_{k|k}}|_{\hat{x}_k, u_k} = \frac{y^{(i)} - y_k}{(x^{(i)} - x_k)^2 + (y^{(i)} - y_k)^2} \\
\frac{\partial \phi^{(i)}_k}{\partial y_{k|k}}|_{\hat{x}_k, u_k} = \frac{- (x^{(i)} - x_k)}{(x^{(i)} - x_k)^2 + (y^{(i)} - y_k)^2} \\
\frac{\partial \phi^{(i)}_k}{\partial \theta_{k|k}}|_{\hat{x}_k, u_k} = -1
\]
1. Compute the predicted state of the robot’s pose using the robot motion model (Equations 3.2–3.4) (i.e. “dead reckoning”) without noise:

\[ \hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k) \]  

(3.17)

2. Compute the residual (error) between the predicted measurement and the received measurement:

\[ \tilde{y}_k = Z_k - h(\hat{x}_{k|k-1}, m) \]

3. Then the following steps are used to iteratively compute the Kalman filter gain

(a) Predict the covariance matrix, \( P_{k|k-1} \) using the previously updated covariance matrix (\( P_{k-1|k-1} \)) and transitional dynamics:

\[ P_{k|k-1} = \hat{F}_k P_{k-1|k-1} \hat{F}_k^T + Q_k \]

(3.18)

where \( Q_k \approx \hat{B}_k D_k \hat{B}_k^T \) as per [5, Chapter 7.4.3] and Equation 3.6.

i. The Jacobian \( \hat{B}_k \) is evaluated: \( \hat{B}_k = \frac{\partial f(\hat{x}_{k-1|k-1}, u_k)}{\partial u_k} \bigg|_{\hat{x}_{k-1|k-1}, u_k} \)

similar to Equation 3.7, but is instead evaluated at the previously updated state estimate.

ii. The Jacobian \( \hat{F}_k \) is evaluated at the previously updated state estimate: \( \hat{F}_k = \frac{\partial f(\hat{x}_{k-1|k-1}, u_k)}{\partial x_{k-1}} \bigg|_{\hat{x}_{k-1|k-1}, u_k} \).

(b) Update the residual covariance matrix \( S_k \):

\[ S_k = \hat{H}_k P_{k|k-1} \hat{H}_k^T + R \]

i. The Jacobian \( \hat{H}_k \) is evaluated at the current predicted state:

\[ \hat{H}_k = \frac{\partial h(\hat{x}_k, m, u_k)}{\partial x_k} \bigg|_{\hat{x}_{k|k-1}, u_k} \].

(c) Calculate the Kalman filter gain:

\[ K_k = P_{k|k-1} \hat{H}_k^T S_k^{-1} \]
3.5. EXTENDED KALMAN FILTER (EKF)

4. With the calculated Kalman filter gain ($K_k$) and residual ($\tilde{y}_k$), the predicted state ($\hat{x}_{k|k-1}$) is **updated** by a correction term which is the Kalman gain multiplied by the residual:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$ (3.19)

5. Finally, update the predicted covariance ($P_{k|k-1}$) and prepare for the next iteration:

$$P_{k|k} = \left( I_{3\times3} - K_k \hat{H}_k \right) P_{k|k-1}$$ (3.20)

### 3.5.1 Simulation Results

Due to the random noise in the system affecting the velocity motion model, Doppler-azimuth data and initial pose, the simulation is repeated $T$ number of times (a “Monte Carlo” simulation) and the average result is reported. This is to reduce the effects of outlier trials from biasing the conclusion of the study.

The root mean square error (RMSE) is estimated by averaging across the Monte Carlo trials:

$$\varepsilon_x(k) = \sqrt{\frac{\sum_{\text{trial}=1}^{T}(\hat{x}^{(\text{trial})}_k - x^{(\text{trial})}_k)^2}{T}}$$

$$\varepsilon_y(k) = \sqrt{\frac{\sum_{\text{trial}=1}^{T}(\hat{y}^{(\text{trial})}_k - y^{(\text{trial})}_k)^2}{T}}$$ (3.21)

$$\varepsilon_\theta(k) = \sqrt{\frac{\sum_{\text{trial}=1}^{T}(\hat{\theta}^{(\text{trial})}_k - \theta^{(\text{trial})}_k)^2}{T}}$$

In the Monte Carlo simulations presented by Guan et al. [6], $T = 50$ trials were used with a sampling interval selected to be $\Delta t = 0.1$ seconds. The robot is driven by a constant control input signal of $u_k = \left[ 0.1 \text{ m/s, } \frac{4\pi}{180} \text{ rad/s} \right]^T$, $\forall k$. The robot’s motion noise parameters for $D_k$ in Equation 3.5 were selected as $\gamma_1 = \gamma_2 = 0.06$ and $\gamma_3 = \gamma_4 = 0.0025$. The Doppler radar noise parameters from Equation 3.11 were selected as $\sigma_d = 0.1$ Hz and $\sigma_\phi = \frac{\pi}{180}$ radians. The robot’s initial pose, $x_0$, is Gaussian distributed around a known mean, $\hat{x}_{0|0}$:

$$\hat{x}_{0|0} = \begin{bmatrix} -1 \text{ m} \\ 1 \text{ m} \\ -\frac{\pi}{4} \text{ rad} \end{bmatrix}$$
and initial covariance, \( P_{0|0} \):

\[
P_{0|0} = \begin{bmatrix}
1.5^2 & 0 & 0 \\
0 & 1.5^2 & 0 \\
0 & 0 & \left(\frac{20\pi}{180}\right)^2
\end{bmatrix}
\]

That is, for every Monte Carlo trial, the initial pose was drawn from the distribution:

\[
x_0 \sim \mathcal{N}(\hat{x}_{0|0}, P_{0|0})
\]  \hspace{1cm} (3.22)

The EKF is also initialised with the known mean and covariance. The conditional information matrix used to calculate the CRLB, is initially equal to the inverse of the initial covariance, as per Equation 3.16.

The simulations results from Guan et al. [6] will now be discussed. The environments investigated in the simulations consist of one, two, three, five or seven landmarks represented by the red diamonds. The map containing \( M = 7 \) landmarks is:

\[
m = \begin{bmatrix}
5 & 0 & -5 & -2 & 4 & 2 & 1 \\
5 & -5 & -3 & 2 & -5 & -1 & 4
\end{bmatrix}^T
\]

The robot equipped with velocity control input data and a Doppler radar moved according to the trajectory represented by the green line (“Actual Trajectory”). The trajectory estimated by the velocity motion model (red dashed line) was initialised at the mean \( \hat{x}_{0|0} \), and the trajectory was created by \( u_k \) without noise. Without using any exteroceptive data, the velocity motion model cannot correct for any initial uncertainty or motion noise, as it is just a form of dead-reckoning. Despite having the same initial estimate as the velocity motion model, the EKF (blue line: “EKF Trajectory Estimate”) is able to estimate the actual robot pose much better because it uses the exteroceptive data from the Doppler radar to update the estimates from the proprioceptive velocity motion model. Figure 3.2 shows one trial of the simulation with seven landmarks.
3.5. EXTENDED KALMAN FILTER (EKF)

Figure 3.2: A single realisation of the EKF compared to the velocity motion model.

Relationship between the number of landmarks and the CRLB

Figure 3.3 shows the CRLB as a function of the number of landmarks in the environment. This gives an indication of the theoretically best result that can be achieved given the number of landmarks available in the environment the robot must operate in. $M = 1, 2, 3, 5$ and $M = 7$ landmarks were investigated in simulation. The results show that an environment with more landmarks strictly decreases the average CRLB (all other factors equal). This result is intuitive because there is more information available in the system when there are more landmarks, hence a more accurate estimation (smaller CRLB) is possible. The scenario with one landmark had the poorest result by far. When $M = 1$, this gives one measurement in Doppler shift and one azimuth measurement which is not much information from the measurement to estimate the three-dimensional state. An environment with one more landmark has a very large benefit to the estimation accuracy, with a relatively small
benefit beyond two landmarks.

Figure 3.3: Cramer-Rao lower bound (CRLB) for the number of landmarks $M = 1, 2, 3, 5,$ and 7.
### 3.5. EXTENDED KALMAN FILTER (EKF)

Relationship between the number of landmarks and the RMSE

Figure 3.4 shows the general relationship that a map with more landmarks decreases the average RMSE (all other factors equal) as there are more measurements which can be used by the EKF algorithm. These results are generally consistent with a decreasing CRLB as shown in Figure 3.3, however, the RMSE does not strictly decrease as can be seen when one landmark is increased to two landmarks. This could be due to random noise or outlier results.

Figure 3.4: RMSE for the number of landmarks $M = 1, 2, 3, 5, \text{ and } 7$. 

(a) RMSE in the $x$-axis. (b) RMSE in the $y$-axis. (c) RMSE in the $\theta$-axis.
CHAPTER 3. LOCALISATION USING DOPPLER RADAR

Effect of varying the number of landmarks on the EKF RMSE compared to the velocity motion model RMSE

Figures 3.5-3.9 compare the EKF RMSE against the velocity motion model RMSE against the CRLB for \( M = 1, 2, 3, 5, \) and \( M = 7 \) landmarks. The results show that as the number of landmarks increase, the data available to the EKF increases which eventually make it a superior choice over the velocity motion model estimate. The result is intuitive because the information from additional landmarks can only be utilised by the exteroceptive sensor - the Doppler-azimuth radar, whereas it does not assist the robot’s proprioceptive velocity motion model estimate in any way.

![Graphs showing comparison between EKF RMSE, velocity motion model RMSE, and CRLB](image)

(a) Comparison in the x-axis.  
(b) Comparison in the y-axis.  
(c) Comparison in the \( \theta \)-axis.

Figure 3.5: Comparisons between the CRLB, EKF RMSE, and velocity motion model RMSE for the case \( M = 1 \) landmark.
3.5. EXTENDED KALMAN FILTER (EKF)

Figure 3.6: Comparisons between the CRLB, EKF RMSE, and velocity motion model RMSE for the case $M = 2$ landmark.
Figure 3.7: Comparisons between the CRLB, EKF RMSE, and velocity motion model RMSE for the case $M = 3$ landmark.
3.5. EXTENDED KALMAN FILTER (EKF)

Figure 3.8: Comparisons between the CRLB, EKF RMSE, and velocity motion model RMSE for the case $M = 5$ landmark.
(a) Comparison in the x-axis.  

(b) Comparison in the y-axis.  

(c) Comparison in the $\theta$-axis.

Figure 3.9: Comparisons between the CRLB, EKF RMSE, and velocity motion model RMSE for the case $M = 7$ landmark.
3.5. EXTENDED KALMAN FILTER (EKF)

Relationship between Doppler uncertainty and the CRLB

Figure 3.10 shows the effect on the CRLB of varying the uncertainty (noise) of the Doppler-shift measurement ($\sigma_d$). Increasing uncertainty of the Doppler-shift measurement should increase the CRLB (all other factors equal) as there is more noise/less information available in the system. The parameter $\sigma_d$ is varied from 0.1 Hz to 100 Hz to 100 kHz. The Doppler uncertainty was investigated as it may give insight into the quality of Doppler radar required for a localisation application, as well as an indication of the theoretically best results that can be achieved. As the Doppler noise becomes very large, the Doppler-azimuth radar should be performing as if it were an azimuth-only radar since the Doppler signal is no longer useful.

Figure 3.10: Cramer-Rao lower bound (CRLB) for variations of uncertainty on the Doppler-shift measurement. There are $M = 7$ landmarks.
CHAPTER 3. LOCALISATION USING DOPPLER RADAR

Relationship between azimuth uncertainty and the CRLB

Figure 3.11 shows the effect on the CRLB of varying the uncertainty (noise) of the azimuth measurement ($\sigma_\phi$). Increasing uncertainty of the azimuth measurement should increase the CRLB (all other factors equal) as there is more noise/less information available in the system. $\sigma_\phi$ is varied from $\frac{\pi}{180}$ to $\frac{20\pi}{180}$ to $\frac{50\pi}{180}$. The azimuth uncertainty was investigated as it may give insight into the number of Doppler radars required in the radar array for a localisation application. Having a higher number of radars in the array should decrease the azimuth uncertainty. As the azimuth noise becomes very large, the Doppler-azimuth radar should be performing as if it were a Doppler-only radar since the azimuth readings are no longer useful.

Figure 3.11: Cramer-Rao lower bound (CRLB) for variations of uncertainty on the azimuth measurement. There are $M = 7$ landmarks.
3.5.2 Simulation Results: Varying Landmarks

Guan et al. [9] also investigated the effect of varying the number of landmarks on the final CRLB and RMSE. The number of landmarks, $M$, was varied from one landmark up to seven landmarks with each landmark represented by a red diamond. The landmarks are positioned as follows:

The map containing $M = 1$ landmarks:

$$m = \begin{bmatrix} 1 \\ 4 \end{bmatrix}^T$$

The map containing $M = 2$ landmarks:

$$m = \begin{bmatrix} 2 & 1 \\ -1 & 4 \end{bmatrix}^T$$

The map containing $M = 3$ landmarks:

$$m = \begin{bmatrix} 4 & 2 & 1 \\ -5 & -1 & 4 \end{bmatrix}^T$$

The map containing $M = 4$ landmarks:

$$m = \begin{bmatrix} -2 & 4 & 2 & 1 \\ 2 & -5 & -1 & 4 \end{bmatrix}^T$$

The map containing $M = 5$ landmarks:

$$m = \begin{bmatrix} -2 & 4 & 2 & 1 & 3 \\ 2 & -5 & -1 & 4 & 3 \end{bmatrix}^T$$

The map containing $M = 6$ landmarks:

$$m = \begin{bmatrix} -2 & 4 & 2 & 1 & 3 & -4 \\ 2 & -5 & -1 & 4 & 3 & 0 \end{bmatrix}^T$$

The map containing $M = 7$ landmarks:

$$m = \begin{bmatrix} -2 & 4 & 2 & 1 & 3 & -4 & 6 \\ 2 & -5 & -1 & 4 & 3 & 0 & 0 \end{bmatrix}^T$$
The simulation parameters, which are identical to the parameters used in the simulations in Chapter 3.5.1, are repeated here for the reader’s convenience. Each scenario was repeated $T = 50$ times to form the Monte Carlo analysis with the sampling interval $\Delta t = 0.1$ seconds. Each simulation lasted for 80 seconds. The Doppler-azimuth radar noise parameters from Equation 3.11 were chosen $\sigma_d = 0.1$ Hz and $\sigma_\phi = \frac{1\pi}{180}$ radians. The robot’s motion was actuated by a constant $u_k = [0.1 \text{ m/s}, \frac{4\pi}{180} \text{ rad/s}]^T$, $\forall k$. The motion noise parameters in $D_k$ in Equation 3.5 were selected as $\gamma_1 = \gamma_2 = 0.06$ and $\gamma_3 = \gamma_4 = 0.0025$.

For every Monte Carlo trial, the initial pose of the robot, $x_0$, was drawn from the distribution:

$$x_0 \sim \mathcal{N}(\hat{x}_{0|0}, P_{0|0})$$

With the initial mean equal to:

$$\hat{x}_{0|0} = \begin{bmatrix} -1 \text{ m} \\ 1 \text{ m} \\ \frac{3\pi}{4} \text{ rad} \end{bmatrix}$$

and the initial covariance equal to:

$$P_{0|0} = \begin{bmatrix} 1.5^2 & 0 & 0 \\ 0 & 1.5^2 & 0 \\ 0 & 0 & (\frac{20}{180})^2 \end{bmatrix}$$

The EKF is initialised with the initial mean and covariance as this information is assumed to be known.

The results from Guan et al. 9 are presented:
Figure 3.12: A single realisation of the EKF with $M = 7$ landmarks.
Figure 3.13: CRLB at the final discrete-time index $k = 800$ for a varying number of landmarks.
3.5. EXTENDED KALMAN FILTER (EKF)

Figure 3.14: RMSE at the final discrete-time index $k = 800$ for a varying number of landmarks.
The results show that the final CRLB is strictly decreasing as the number of landmarks increases which is expected as the addition of more landmarks strictly adds more information i.e. strictly reduces uncertainty. The CRLB has the largest decrease in magnitude when the second landmark is added but the addition of further landmarks show a type of diminishing return to the reduction in uncertainty.

In comparison, the effect on the reduction of RMSE of the EKF is less drastic. Increasing the number of landmarks does not guarantee the reduction of RMSE of the EKF possibly due to random noise. The addition of landmarks five, six and seven may not have contributed much to reduction in RMSE possibly due to some information redundancy as their positions did not add much azimuth variations compared to pre-existing landmarks one to four.

3.6 Sequential Importance Resampling (SIR) Particle Filter

The SIR particle filter (PF) has three main steps: predict, update and re-sample. These steps are also described by Choi et al. [143, p. 2] and more general details can be found in Chapter 2.5.2. The following procedure refers to Algorithm 1 and the number of particles \( N \) is fixed.

Initially, draw the particles from the initial distribution: \( p_0(x) \), if it is known. Otherwise, uniformly distributed particles covering the state-space is a common choice for initialising the particle filter.

In Algorithm 1, the robot’s pose is predicted by propagating each particle through the robot’s motion model as per Equations 3.2 and 3.4, given the known control input signal. This is computed on Line 3.

Then, the unnormalised weight of each particle is calculated by the likelihood of the received measurement,

\[
\hat{w}_k^{[i]} = \psi(Z_k^{[i]}, \mathbf{x}_k^{[i]}, m, u_k)
\]

as per the update step. To do so, a predicted measurement from the point-of-view of each particle is created, i.e. \( h(x_k^{[i]}, m, u_k), \forall i \), and a subsequent likelihood function is formed around this prediction. This likelihood formed around the predicted measurement is then evaluated at the value of the received measurement. Since the work in this chapter assumes the mea-
3.6. **SEQUENTIAL IMPORTANCE RESAMPLING (SIR) PARTICLE FILTER**

Measurement noise is from a multivariate Gaussian distribution, the likelihood for each particle would be:

\[
\tilde{w}_k^{[i]} = \psi(Z_k | x_k^{[i]}, m, u_k) = \mathcal{N}(Z_k; h(x_k^{[i]}, m, u_k), R_k)
\]

That is, the multivariate Gaussian has a mean equal to the predicted measurement and is evaluated at the point of the received measurement, \((Z_k)\). Alternatively under the multivariate Gaussian assumption, this can also be done by calculating the residual between the received measurement and the predicted measurement:

\[
\tilde{y}_k^{[i]} = Z_k - h(x_k^{[i]}, m, u_k)
\]

Then the unnormalised weight is equal to the zero-mean multivariate Gaussian evaluated at the point of the residual:

\[
\tilde{w}_k^{[i]} = \psi(Z_k | x_k^{[i]}, m, u_k) = \mathcal{N}(\tilde{y}_k^{[i]}, 0_{2L_k \times 1}, R_k)
\]

This step occurs on Line 7 of Algorithm 1.

Once the particles have completed the predict and update steps, resampling is performed where the particles are sampled with replacement in proportion to their weight. This occurs on Line 11 of Algorithm 1. Statistically, highly weighted particles are duplicated multiple times and lowly weighted particles can disappear. For simplicity, it is assumed that the particle filter resamples at every discrete-time index \(k\) although other popular implementations resample when the effective sample size is below a threshold \([92], [93]\). This is also discussed in Chapter 2.5.2. Following resampling, the weights are already normalised.

The SIR particle filter is presented in Algorithm 1.
Algorithm 1 SIR Particle Filter

1: **Input:** \( \{ x^{[i]}_{k-1}, w^{[i]}_{k-1} \}_{i=1}^{N}, u_k, Z_k \)
2: **for** \( i = 1 : N \) **do**
3: \( x^{[i]}_k \sim \pi(x^{[i]}_{k-1}, u_k) \) \( \triangleright \) Predict: Apply noisy motion model to each particle.
4: \( \tilde{w}^{[i]}_k = \psi(Z_k|x^{[i]}_k, m, u_k) \) \( \triangleright \) Update: The weight of each particle is equal to the likelihood function evaluated at the measurement.
5: **end for**
6: Normalise the weights: \( w^{[i]}_k = \frac{\tilde{w}^{[i]}_k}{\sum_{j=1}^{N} \tilde{w}^{[j]}_k} \)
7: Calculate the weighted mean of the particle filter.
8: Resample: \( \{ x^{[i]}_k, w^{[i]}_k \}_{i=1}^{N} \rightarrow \{ x^{[i]}_k, \frac{1}{N} \}_{i=1}^{N} \)
9: **Output:** \( \{ x^{[i]}_k, \frac{1}{N} \}_{i=1}^{N} \)

Following the initialisation of the particles and uniform weights, the mean of the SIR particle filter in the \( x \) and \( y \)-axes were calculated according to the simple mean:

\[
\hat{x}_k = \frac{1}{N} \sum_{i=1}^{N} x^{[i]}_k \quad (3.23)
\]

\[
\hat{y}_k = \frac{1}{N} \sum_{i=1}^{N} y^{[i]}_k \quad (3.24)
\]

How should the (unweighted) mean of the heading \( \theta \) be calculated? Consider a basic example: what is the mean of the angles (in degrees, for simplicity): \(+20^\circ\) and \(+340^\circ\)? The reader would know that \(+340^\circ\) must be wrapped “correctly” to convert it to \(-20^\circ\) which leads to the conclusion that the average of the two quantities is \(0^\circ\). Which wrapping system would be used if three angles had to be averaged? Which wrapping system would be used if the heading of all \( N \) particles must be wrapped and averaged? Hence, the study of the mean of circular/periodic variables. This is explained by Olson [144, p. 2], Jammalamadaka and SenGupta [145, Chapter 1.3.1], and Bishop [146, Chapter 2.3.8]. The (unweighted) mean of the heading \( \theta \) was calculated:
\[ \hat{\theta}_k = \text{atan2} \left( \frac{\sum_{i=1}^{N} \sin \theta[i]}{\sum_{i=1}^{N} \cos \theta[i]} \right) \] (3.25)

which is a typical way to calculate the mean of circular/periodic variables. The function: \(\text{atan2}(\cdot)\), represents a four-quadrant arctangent function \([144, p. 2]\). Note that a number of other sources also include additional \(\frac{1}{N}\) terms, see: \([147, pp. 54-55], [148, p. 15], [149, p. 1363], [150, p. 2501]\).

The weighted mean of the SIR particle filter was calculated prior to re-sampling. The weighted mean for the \(x\) and \(y\)-axes were calculated as per Equation 2.9 and Equation 2.10. For convenience, they are restated:

\[ \hat{x}_k = \sum_{i=1}^{N} w[i] \cdot x[i] \] (3.26)

\[ \hat{y}_k = \sum_{i=1}^{N} w[i] \cdot y[i] \] (3.27)

However, for the \(\theta\)-axis, the weighted mean is instead calculated:

\[ \hat{\theta}_k = \text{atan2} \left( \frac{\sum_{i=1}^{N} w[i] \cdot \sin \theta[i]}{\sum_{i=1}^{N} w[i] \cdot \cos \theta[i]} \right) \] (3.28)

The ad-hoc formulation of the weighted angular mean in Equation 3.28 was based on the contributions on the stackoverflow.com website by a user named “Chris” \([151]\). The intuition behind this formula is supported by Yamartino \([149, p. 1363]\), and the United States Environmental Protection Agency - Office of Air Quality Planning and Standards \([147, pp. 53-54]\) where the mean angle (wind direction) is calculated by weighting each angle by its magnitude (wind speed).

### 3.7 Exact Daum-Huang (EDH) Particle Flow Particle Filter

One of the problems faced by particle filters is called weight “degeneracy” explained by Li et al. \([93]\). This problem occurs when the majority of particles drawn from the prior distribution have negligible weight. In the context
of mobile robot localisation, the prior distribution is typically determined by a probabilistic model of how the robot moves, whereas the weight of each particle is determined by the likelihood of the sensor measurements. The likelihood of the sensor measurement is usually a relatively narrow likelihood in comparison to the wider, cruder, prior distribution based on the motion model. For this reason, only a few particles in the prior distribution tend to have a high weight. This is the degeneracy problem, and a consequence is that the posterior probability distribution is represented by only one, or a few, particles, with the rest having zero, or negligible weight. This is not a good way to represent any probability distribution. The Exact Daum-Huang (EDH) particle flow particle filter (PFPF) is an algorithm designed to mitigate this problem. For further reading, see the references by Daum and Huang [111], Daum et al. [112], and Li and Coates [110].

3.7.1 Overview of the EDH Particle Flow Particle Filter

The particle flow particle filter has an additional step whereby the particles “flow” from the prior distribution to the posterior distribution with some dependency on the likelihood and the received measurements. This causes the particles to be more evenly distributed around the narrow posterior distribution caused by the narrow measurement likelihood. As a result of this, the particle weights are more balanced, the posterior is better represented and this therefore mitigates the particle filter degeneracy problem.

Assuming the measurement model for the Doppler radar in Equation 3.14 (therefore, the likelihood function $\psi(\cdot)$ has a dependency on the control input $u_k$), the unnormalised conditional posterior probability density is computed using Bayes’ rule - adapted to be applied to the mobile robot localisation problem [5, Chapter 2.3.4, Chapter 4.3.1]:

$$p(x_k|Z_{1:k}, u_{1:k}) = \psi(Z_k|x_k, m, u_k) \cdot p(x_k|Z_{1:k-1}, u_{1:k})$$

Equation 3.29 is much like Equation 2.2 without normalisation.

Standard particle filters implement Bayes’ rule as the point-wise multiplication of two functions: First, the prior distribution: $p(x_k|Z_{1:k-1}, u_{1:k})$, is approximated by the particle set after the particles have been propagated through the noisy motion model in Section 3.2.1. Second, the approximated
prior distribution is weighted by the likelihood of the current measurement:
\[ \psi(Z_k|x, m, u_k) \]
which gives a weight to each particle.

Let \( g(x) = p(x_k|Z_{1:k-1}, u_{1:k}) \) be the prior density and let \( h(x) = \psi(Z_k|x, m, u_k) \) be the likelihood of the measurement. Let \( p(x) = p(x_k|Z_{1:k}, u_{1:k}) \) be the posterior. When this shorthand notation is applied to Equation 3.29, the result is:

\[
p(x) = h(x) \cdot g(x)
\] (3.30)

Then, take the logarithm of both sides of the posterior density relationship in Equation 3.30 which results in:

\[
\log[p(x)] = \log[g(x)] + \log[h(x)]
\] (3.31)

Then, turn this into a homotopy function (a function that can continuously change to another function) by introducing a variable \( \Lambda \in [0, 1] \):

\[
\log[p(x)] = \log[g(x)] + \Lambda \log[h(x)]
\]

It can be seen that when \( \Lambda = 0 \), then \( \log[p(x)] = \log[g(x)] \). Therefore, \( p(x) = g(x) \) which is the prior distribution. Conversely, when \( \Lambda = 1 \), then \( \log[p(x)] = \log[g(x)] + \log[h(x)] \) which is the relationship in Equation 3.31 which was simply the logarithm of the posterior distribution in Equation 3.29.

Therefore, when \( \Lambda \) is changing from 0 to 1, the function \( p(x) \) is transforming from the prior distribution to the posterior distribution. In the literature, \( \Lambda \) is referred to as “pseudo-time” because it is changing between every sample of actual time which elapses with discrete-time index \( k \).

The characteristics of pseudo-time are defined by the parameters \( N_\Lambda \), which determines the number of steps in pseudo-time, and a step size interval of \( \Delta \Lambda \) between pseudo-time steps. For simplicity, it is assumed that the step size interval is constant, however this is not the only choice, as Li and Coates [110, p. 4108] try intervals which are exponentially spaced.

How should each particle move as pseudo-time \( \Lambda \) is changing?

Define:

\[
\frac{dx}{d\Lambda} = f(x, \Lambda)
\]

Using the chain rule and the Fokker-Planck Equation with zero process noise, according to Daum and Huang [111] and Daum et al. [112], the following partial differential equation is produced:
The solution to Equation 3.32 can lead to a number of different particle flow filters depending on the constraints, most of which will not be analytically tractable. In the special case where the predictive posterior and the likelihood are both Gaussian distributions, and the measurement model is linear (or else, the non-linear model has to be linearised), the flow trajectory results in the Exact Daum-Huang (EDH) particle flow particle filter [110].

In the scenario where the prior and the likelihood are Gaussian, the solution for the particle “flow” with respect to Λ is:

\[
\frac{dx}{d\Lambda} = A(\Lambda)x + b(\Lambda) \tag{3.33}
\]

So it can be seen that as pseudo-time Λ changes from 0 to 1, the particles are induced to flow/move in between every sample of actual time incremented by discrete-time index \( k \).

Li and Coates [110, p. 4105] present the calculations for the particle flow particle filter’s flow parameters \((A(\Lambda) \text{ and } b)\) given a non-linear measurement model, which are required since this problem assumes the use of a Doppler-azimuth radar as the sensor, which has a non-linear measurement model. The parameters \( A(\Lambda) \text{ and } b \) in Equation 3.33 are:

\[
A(\Lambda) = -\frac{1}{2} P_{k|k-1} H^T (\Lambda H P_{k|k-1} H^T + R)^{-1} H \tag{3.34}
\]

where \( H = \frac{\partial h}{\partial x} \bigg|_{x=\eta} \). The function \( h \) is the same as described in Equation 3.10. The predicted covariance matrix \( P_{k|k-1} \) is calculated using the EKF Equation 3.18. The measurement noise covariance matrix \( R_k \) is also required. The variable \( \eta \) is initially equal to the predicted mean following noiseless motion, \( \bar{x} \), however, unlike \( \bar{x} \), \( \eta \) will later flow in pseudo-time according to the flow parameters.

The flow parameter \( b \) is calculated:

\[
b(\Lambda) = \{I + 2\Lambda A(\Lambda)\} \left[\{I + \Lambda A(\Lambda)\} P_{k|k-1} H^T R^{-1} \{Z_k - a(\Lambda)\} + A(\Lambda)\bar{x}\right] \tag{3.35}
\]

Where

\[
a(\Lambda) = h(\eta, m, u_k) - H\eta \tag{3.36}
\]
3.7. EXACT DAU-M-HUANG (EDH) PARTICLE FLOW PARTICLE FILTER

The calculations in Equation 3.36 and Equation 3.35 introduce the residual term between the received measurement $Z_k$, and the predicted measurement $h(\cdot)$ evaluated at the “flowing” mean of the particle set: $\bar{\eta}$. This step is similar to the EKF and the particle filter where the current measurement is compared against a predicted measurement.

3.7.2 Implementation

The EDH particle flow particle filter described by Li and Coates [110, p. 4108] is presented in Algorithm 2. Referring to Algorithm 2, a description of the key steps required to implement the EDH particle filter follows:

1. Line 6 applies the EKF predict equations to calculate a predicted mean, $\hat{x}_{k|k-1}$ and covariance, $P_{k|k-1}$ from Equations 3.17 and 3.18 respectively.

2. Line 9 applies the motion model with noise from Equation 3.1 to each particle: $x_{k}^{[i]} = f(x_{k-1}^{[i]}, u_k) + e_k$.

3. Line 10 introduces a new variable $\eta_{1}^{[i]}$ which represent “intermediate” particles that will flow in pseudo-time. Initially, $\eta_{1}^{[i]} = x_{k}^{[i]}$, which is the starting point of the flowing particle.

4. Line 13 applies the motion model without noise to the mean at $k - 1$ ($\bar{x}_{k-1|k-1}$) and defines this as the variable $\bar{x}$.

5. Line 15 defines the mean of the intermediate distribution, $\bar{\eta} = \bar{x}$. The mean of the intermediate distribution, $\bar{\eta}$ will flow in pseudo-time. The variable $\bar{x}$ is used to calculate the flow parameter $b(\Lambda)$ in Equation 3.35 occurring on Line 20.

6. Line 18 begins the loop in pseudo-time. That is, for every discrete-time index $k$, pseudo-time $\Lambda \in [0, 1]$ is elapsing from 0 to 1 with $N_{\Lambda}$ steps, with a step size interval of $\Delta \Lambda$. During this pseudo-time loop from $j = 1 : N_{\Lambda}$, the intermediate particles will be flowing according to the flow parameters.

7. On Line 20 calculate the flow parameters $A(\Lambda)$ and $b(\Lambda)$ according to Equations 3.34 and 3.35. Then, apply the calculated flow to the flowing
8. Calculate the unnormalised weight for each particle on Line 27.

9. Line 29 updates the particle set so that the particles, \( x_i^k \), are equal to the pseudo-time, flowing particles, \( \eta_i^1 \), after the flow has been completed.

10. In the situation where none of the particles have a weight, which occurs if the particles have been proposed poorly, the weights are reinitialised and the mean is set to the average of the particles as per Equations 3.23-3.25. If the particles have a non-zero weight as it typically should, the weights are normalised, the mean is equal to the weighted mean of the particles as per Equations 3.26-3.28 and a resample is performed. This occurs on Lines 30-39.

11. Finally, apply the EKF update step on Line 40 to calculate the updated covariance matrix \( P_{k|k} \). This requires use of the predicted mean \( \hat{x}_{k|k-1} \) as per the EKF update step in Equation 3.20. The updated mean, \( \hat{x}_{k|k} \), has already been calculated as the weighted mean of the particle set.
Algorithm 2 EDH Particle Flow Particle Filter

1: Draw particle set \( \left\{ x_0^{[i]} \right\}_{i=1}^N \) from the initial prior distribution \( p_0(x_0) \). The mean and covariance, respectively, of \( p_0 \), are \( (\hat{x}_0, P_0) \).

2: Initialise particle weights uniformly: \( \left\{ w_0^{[i]} \right\}_{i=1}^N = \frac{1}{N} \)

3: Input: \( \left\{ x_{k-1}^{[i]}, w_{k-1}^{[i]} \right\}_{i=1}^N, u_k, Z_k, \hat{x}_{k-1|k-1}, P_{k-1|k-1} \)

4: for \( k = 1 : N_{\text{sim}} \) do

5: Use Equations 3.17 and 3.18 to make a prediction for the mean and covariance: \( (\hat{x}_{k-1|k-1}, P_{k-1|k-1}) \rightarrow (\hat{x}_{k|k-1}, P_{k|k-1}) \)

6: \( \triangleright \) The predicted mean will be required for Line 40

7: for \( i = 1 : N \) do

8: Apply the noisy motion model in Equation 3.1 to each particle:

9: \[ x_k^{[i]} = f(x_{k-1}^{[i]}, u_k) + e_k. \]

10: Let \( \eta_1^{[i]} = x_k^{[i]} \)

11: \( \triangleright \) \( x_k^{[i]} \) will be stationary, but it is used to evaluate the transitional probability on Line 27.

12: \( \eta_1^{[i]} \) will be flowing according to the flow parameters on Line 23.

13: end for

14: end for

15: Use the motion model in Equation 3.1 without noise, to predict the mean of the particle set: \( \bar{x} = f(\hat{x}_{k-1|k-1}, u_k) \)

16: \( \bar{\eta} = \bar{x}. \) (Note: \( \bar{x} \) will be stationary at discrete-time index \( k \), but \( \bar{\eta} \) will be flowing according to the flow parameters on Line 21).

17: \( \Lambda = 0 \)

18: for \( j = 1 : N_\Lambda \) do

19: \( \Lambda = \Lambda + \Delta \Lambda \)

20: Calculate the flow parameters \( A(\Lambda) \) and \( b(\Lambda) \) according to Equations 3.34 and 3.35

21: Migrate \( \bar{\eta} \) according to the flow parameters: \( \bar{\eta} = \bar{\eta} + \Delta \Lambda (A(\Lambda)\bar{\eta} + b(\Lambda)) \)

22: for \( i = 1 : N \) do

23: Migrate each intermediate particle according to the flow parameters: \( \eta_1^{[i]} = \eta_1^{[i]} + \Delta \Lambda \left\{ A(\Lambda)\eta_1^{[i]} + b(\Lambda) \right\} \)

24: end for

25: end for
for $i = 1 : N$ do

Calculate particle weight: 

$$w_k^i = \frac{w_{k-1}^i \cdot \pi(\eta_1^i | x_{k-1}^i, u_k) \cdot \psi(Z_k | \eta_1^i, m, u_k)}{\pi(x_k^i | x_{k-1}^i, u_k)}$$

end for

Update particle set: $x_k^i = \eta_1^i$

if $\sum_{i=1}^N \tilde{w}_k^i = 0$ then \>

In the situation where none of the particles have a weight, reinitialise the weights uniformly and calculate the mean of the particles as the simple mean.

for $i = 1 : N$ do

$w_k^i = \frac{1}{N}$

$\hat{x}_{k|k} = \text{simple mean, as per Equations 3.23-3.25}$

end for

else

for $i = 1 : N$ do

Normalise the weights: 

$$w_k^i = \frac{\tilde{w}_k^i}{\sum_{j=1}^N \tilde{w}_k^j}$$

$\hat{x}_{k|k} = \text{weighted mean, as per Equations 3.26-3.28}$

Resample: 

$$\{x_k^i, w_k^i\}_{i=1}^N \rightarrow \{x_k^i, \frac{1}{N}\}$$

Apply the EKF covariance update using Equation 3.20: 

$(\hat{x}_{k|k-1}, P_{k|k-1}) \rightarrow (\hat{x}_{k|k}, P_{k|k})$. \>

Note: The updated mean $\hat{x}_{k|k}$ has already been defined as the weighted mean of the particle set.

end for

end if

end for

Output: 

$$\{x_k^i, w_k^i\}_{i=1}^N, \hat{x}_{k|k}, P_{k|k}$$
3.7. EXACT DAUM-HUANG (EDH) PARTICLE FLOW PARTICLE FILTER

3.7.3 Simulation Results and Comparative Studies

A Monte Carlo simulation was performed with $T = 100$ trials performed. Each trial lasted for $N_{\text{sim}} = 40$ samples with a sampling interval of $\Delta t = 1$. Three algorithms were compared in this study: EKF, SIR particle filter and the EDH particle flow particle filter. To evaluate the estimation accuracy of the algorithm, the RMSE was calculated as per Equation 3.21. Failed trials were counted and removed from the RMSE calculations to reduce the effect of outlier results. A trial is defined as a failure if any of the final 10 MMSE estimates exceed 1.5 meters from the true pose. The average computation time included failed trials in the calculation. The simulation was completed in MATLAB on an Intel Core i5-6600 3.30GHz CPU with 32 GB RAM. Parallel computing was used for parts of the SIR particle filter as well as the EDH particle flow particle filter.

For every Monte Carlo trial, the initial pose of the robot, $x_0$, was drawn from the distribution:

$$x_0 \sim N(\hat{x}_{0|0}, P_{0|0})$$

where the initial mean was:

$$\hat{x}_{0|0} = \begin{bmatrix} 0 \text{ m} \\ 0 \text{ m} \\ 0 \text{ rad} \end{bmatrix}$$

and the initial covariance was:

$$P_{0|0} = \begin{bmatrix} 3^2 & 0 & 0 \\ 0 & 3^2 & 0 \\ 0 & 0 & (\frac{20\pi}{180})^2 \end{bmatrix}$$

A constant control signal was applied to actuate the robot’s motion: $u_k = [0.2, \frac{4\pi}{180}]^T, \forall k$. The motion noise was modelled as a multivariate Gaussian with zero mean and constant covariance matrix:

$$Q_k = \begin{bmatrix} 0.1^2 & 0 & 0 \\ 0 & 0.1^2 & 0 \\ 0 & 0 & (\frac{15\pi}{180})^2 \end{bmatrix}, \forall k$$

The Doppler radar noise had a standard deviation of $\sigma_d = 0.2$ and the standard deviation of the azimuth was $\sigma_\phi = \frac{5\pi}{180}$. 
CHAPTER 3. LOCALISATION USING DOPPLER RADAR

The implementation of the EKF was akin to the implementation described in Chapter 3.5. The EKF was initialised at the initial mean ($\hat{x}_{0|0}$) and covariance ($P_{0|0}$) which are assumed to be known.

The SIR particle filter required $N = 10,000$ particles.

The EDH particle flow particle filter used a constant pseudo-time step size of $\Delta \Lambda = 0.1$, therefore leading to $N_\Lambda = \frac{1}{0.1} = 10$ pseudo-time steps since $\Lambda \in [0, 1]$. The EDH particle flow particle filter used $N = 200$ particles.

The SIR particle filter and EDH particle flow particle filter had particles initialised according to the known distribution of the initial pose. That is:

$$x_k^{[i]} \sim N(\hat{x}_{0|0}, P_{0|0}), \forall i$$

In the first scenario, three landmarks ($M = 3$) were placed in the environment, the map being:

$$m = \begin{bmatrix} 2 & 1 & -5 \\ 5 & -8 & 6 \end{bmatrix}^T$$

A single trial of the simulation is shown in Figure 3.15. The RMSE results for this scenario are shown in Figure 3.16. Table 3.1 shows the number of failures, final RMSEs and computation times. The RMSE results show that the SIR particle filter is able to converge at an earlier time compared to the EKF and EDH particle flow particle filter. The SIR particle filter also recorded no failures. All three algorithms achieve a similar steady-state error, with the SIR particle filter recording a slightly lower RMSE when the RMSE in the three dimensions are added together. The EDH particle flow particle filter had the highest RMSE and the most number of failures (three) compared to the EKF with two failures. The computation times show that the EKF is by far, the fastest algorithm. The SIR particle filter is the slowest and the EDH particle flow particle filter is relatively slow compared to the EKF, but is still faster than the SIR particle filter. The SIR particle filter is 4.70 times slower than the EDH particle flow particle filter and 1954.89 times slower than the EKF. The EDH particle flow particle filter is 415.52 times slower than the EKF.
3.7. **EXACT DAUM-HUANG (EDH) PARTICLE FLOW PARTICLE FILTER**

![Diagram](image)

**Figure 3.15:** A single realisation for $M = 3$ landmarks.
CHAPTER 3. LOCALISATION USING DOPPLER RADAR

Figure 3.16: RMSE for $M = 3$ landmarks.
In the second scenario, the number of landmarks was increased to $M = 4$, the map being:

$$
m = \begin{bmatrix} 2 & 1 & -5 & 7 \\ 5 & -8 & 6 & 3 \end{bmatrix}^T
$$

A single trial of the simulation is shown in Figure 3.17. The RMSE results are shown in Figure 3.18 and the final RMSEs, failures and computation times are shown in Table 3.2. The results show that the SIR particle filter converges faster, to a slightly smaller final RMSE compared to the other two algorithms and recorded zero failures. The EDH particle flow particle filter recorded three failures in this scenario which was more than the one failure from the EKF. The SIR particle filter is 5.04 times slower than the EDH particle flow particle filter and 2307.90 times slower than the EKF. The EDH particle flow particle filter is 457.69 times slower than the EKF.
Figure 3.17: A single realisation for $M = 4$ landmarks.
3.7. **EXACT DAUM-HUANG (EDH) PARTICLE FLOW PARTICLE FILTER**

(a) RMSE in the $x$-axis.

(b) RMSE in the $y$-axis.

(c) RMSE in the $\theta$-axis.

Figure 3.18: RMSE for $M = 4$ landmarks.
In the final scenario, the number of landmarks was increased to $M = 5$, the map being:

$$ m = \begin{bmatrix} 2 & 1 & -5 & 7 & -5 \\ 5 & -8 & 6 & 3 & -5 \end{bmatrix}^T $$

A single trial of the simulation is shown in Figure 3.19. The RMSE results are shown in Figure 3.20 and the final RMSEs, failures and computation times are shown in Table 3.3. The results show that the SIR particle filter converges faster, to a slightly smaller final RMSE compared to the other two algorithms. None of the algorithms failed in this scenario. The SIR particle filter is 5.37 times slower than the EDH particle flow particle filter and 2369.16 times slower than the EKF. The EDH particle flow particle filter is 441.57 times slower than the EKF.
Figure 3.19: A single realisation for $M = 5$ landmarks.
CHAPTER 3. LOCALISATION USING DOPPLER RADAR

Figure 3.20: RMSE for $M = 5$ landmarks.
3.8. CONCLUSION AND DISCUSSION

The results for the scenario with five landmarks:

<table>
<thead>
<tr>
<th></th>
<th>EKF</th>
<th>SIR Particle Filter</th>
<th>EDH Particle Flow Particle Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failures per 100 Trials</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Final $\epsilon_x$ (m)</td>
<td>0.1084</td>
<td>0.1051</td>
<td>0.1264</td>
</tr>
<tr>
<td>Final $\epsilon_y$ (m)</td>
<td>0.0939</td>
<td>0.0931</td>
<td>0.1041</td>
</tr>
<tr>
<td>Final $\epsilon_\theta$ (rad)</td>
<td>0.0113</td>
<td>0.0111</td>
<td>0.0116</td>
</tr>
<tr>
<td>Final $(\epsilon_x + \epsilon_y + \epsilon_\theta)$</td>
<td>0.2136</td>
<td>0.2093</td>
<td>0.2421</td>
</tr>
<tr>
<td>Average Simulation Time (sec)</td>
<td>0.0061</td>
<td>14.4519</td>
<td>2.6936</td>
</tr>
</tbody>
</table>

Table 3.3: Scenario with five landmarks.

The results of the three scenarios tentatively indicate that the SIR particle filter does not scale as efficiently to an increasing number of landmarks. This is demonstrated by the SIR particle filter’s increasing absolute computation time, as well as its computation time relative to the EKF and EDH particle flow particle filter, which is trending upwards with the number of landmarks. The EDH particle flow particle filter also requires more absolute computation time for an increasing number of landmarks. The EKF does not seem to have a clear relationship between its computation time and the number of landmarks.

3.8 Conclusion and Discussion

This chapter demonstrates that under a set of limiting assumptions, the mobile robot localisation problem can be solved using a Doppler-azimuth radar and velocity control input signals, in a feature-based map. This chapter shows that there is sufficient information provided by the Doppler-azimuth radar and velocity control input signals by demonstrating a low CRLB. The CRLB was compared against the achieved RMSE of the EKF. The difference between the performance of the EKF and the best achievable accuracy inferred by the CRLB indicate that there are likely better algorithms that can be applied to the scenario over the EKF as there is a noticeable difference between the CRLB and the EKF’s RMSE. This motivates the use of particle filters to solve the problem. It is shown that the SIR particle filter, state-of-the-art EDH particle flow particle filter as well as the EKF, can all be used to solve the localisation problem. The comparative studies indicate that the
SIR particle filter generally has the best estimation quality and least number of failures. However, it requires the most computational resources, by far. In comparison, the EDH particle flow particle filter is significantly faster to run than the SIR particle filter, but its estimation accuracy is worse than both the EKF and the SIR particle filter and it also recorded the highest number of failures. The EKF is by far, the fastest algorithm to use, and it demonstrates satisfactory performance under the assumptions in this chapter.
Chapter 4

Localisation using Doppler Radar: Relaxing the Assumptions

4.1 Introduction

The previous chapter shows that under a set of limiting assumptions, mobile robot localisation can be achieved using a Doppler radar in a feature-based map. The previous assumptions simplistically model the Doppler radar noise as a Gaussian, and also assume that the measurements are somehow automatically associated with the landmark they are returned from. It is also assumed that the robot’s initial pose distribution is known.

This chapter makes the simulation studies more realistic by improving on four of the limiting assumptions: first, landmarks can now be missed by the radar - i.e. a landmark is in the environment but does not generate a measurement as it has been missed; second, false (spurious) detections/“clutter” can occur - i.e. the radar detects a landmark when there is none; third, unknown landmark-measurement associations - i.e. it is uncertain which measurement is being produced from which landmark; and finally, there is no known information about the robot’s initial pose.

A proposed solution to this problem is the random finite set (RFS) particle filter. The RFS particle filter essentially considers all possible hypotheses over the combinations of hits, misses, false detections and landmark-measurement associations (the only hypothesis excluded from the RFS par-
particle filter likelihood is the hypothesis where all landmarks are assumed to be missed). To assist the reader in understanding how these hypotheses are formulated, a number of examples of the list of hypotheses and the resulting likelihood are included in Chapter 4.3.

As mentioned in the literature review, particle filters are computationally very expensive. Application of the RFS particle filter requires the list of hypotheses which grows exponentially with the number of landmarks. It is practically necessary to restrict the number of hypotheses to be evaluated. Murty’s $k$-best assignment algorithm is used to select the $k$-best association hypotheses. This is discussed in Chapter 4.3.3.

To further ease computational requirements, the number of particles can also be adapted by means of KLD-sampling which basically reduces the number of particles as the uncertainty of the particle filter decreases. This will be introduced in Chapter 4.4.1.

The results of this chapter were published to a journal paper by Guan et al. [7].

### 4.2 Motion Model

The robot motion model assumptions are identical to the assumptions used in Chapter 3.2.1, and this will be restated for convenience. With reference to Equation (3.1), the velocity motion model is described by Thrun et al. [5, Chapter 7.4.3, Chapter 5.3.3] where the control input applied from $k - 1$ to $k$ comprises of a translational, as well as a rotational velocity, denoted by: $u_k = [v_k, \omega_k]^T$. The function $f(\cdot)$ from Equation (3.1) is a $3 \times 1$ vector with the following three components that describe the robot’s motion:

\[
\begin{align*}
    f_1(x_{k-1}, u_k) &= x_{k-1} - \frac{v_k}{\omega_k} \left[ \sin \theta_{k-1} - \sin(\theta_{k-1} + \omega_k \Delta t) \right] \\
    f_2(x_{k-1}, u_k) &= y_{k-1} + \frac{v_k}{\omega_k} \left[ \cos \theta_{k-1} - \cos(\theta_{k-1} + \omega_k \Delta t) \right] \\
    f_3(x_{k-1}, u_k) &= \theta_{k-1} + \omega_k \Delta t
\end{align*}
\]

$\Delta t = t_k - t_{k-1}$ is the sampling interval.

Referring to [5, pp. 205-206], the motion noise is originally assumed to be added to the control input $u_k$. However, in order to be compatible with the extended Kalman filter state space assumptions, the motion noise is
4.3. MEASUREMENT MODEL

assumed to be distributed according to a zero-mean, multivariate Gaussian distribution: \( e_k \sim \mathcal{N}(0_{3 \times 1}, Q_k) \) and is added to the motion model \( f(\cdot) \) in the state space, as per Equation 3.1. This therefore requires that the motion noise in the control input space be converted to motion noise in the state space. Let the diagonal matrix:

\[
D_k = \begin{bmatrix}
\gamma_1 v_k^2 + \gamma_2 \omega_k^2 & 0 \\
0 & \gamma_3 v_k^2 + \gamma_4 \omega_k^2
\end{bmatrix}
\]

be the covariance of the noise for the control inputs. \( D_k \) has parameters \( \gamma_1, \gamma_2, \gamma_3, \) and \( \gamma_4 \) which should reflect noise levels.

The approximation of the covariance of the motion noise in state space, is:

\[
Q_k \approx B_k D_k B_k^T
\]

which is an approximate mapping of motion noise from the control input space, to the state space.

The model for clutter is defined as a Poisson point process with statistics that do not vary with time, nor space (for simplicity). The average rate of occurrence of these false detections is denoted by \( \lambda \). When a false detection occurs, the spatial distribution of the false detection is \( c(z_k^{(j)}) \). This means the number of false detections and the value of the false detection are both random.

4.3 Measurement Model

The measurement model must accommodate the possibilities of missed and false/spurious detections (clutter). Therefore, the number of measurements (cardinality) at discrete-time \( k \) must be random. The number of measurements received at discrete-time \( k \) is denoted as \( L_k \).

The transitional density can therefore be expressed as

\[
\pi(x_k|x_{k-1}, u_k) = \mathcal{N}(x_k; f(x_{k-1}, u_k), Q_k)
\]

4.3 Measurement Model

The measurement model must accommodate the possibilities of missed and false/spurious detections (clutter). Therefore, the number of measurements (cardinality) at discrete-time \( k \) must be random. The number of measurements received at discrete-time \( k \) is denoted as \( L_k \).

The model for clutter is defined as a Poisson point process with statistics that do not vary with time, nor space (for simplicity). The average rate of occurrence of these false detections is denoted by \( \lambda \). When a false detection occurs, the spatial distribution of the false detection is \( c(z_k^{(j)}) \). This means the number of false detections and the value of the false detection are both random.
Missed detections are modelled using a probability of missing a landmark. The probability of detecting landmark \( j \) is denoted as \( P_d(m^{(j)}) \leq 1 \). In this work, the probability of detection is modelled as:

\[
P_d(m^{(j)}) = e^{-\beta \times D_j^2}
\] (4.7)

Where \( \beta \) is a constant and \( D_j \) is the distance between the robot and landmark \( j \). As the distance \( D_j \) between the robot and the landmark increases, the probability of detection of the landmark decreases. When a landmark is detected, the noise is assumed to be zero-mean, Gaussian. In the event that a landmark has been missed, no measurement is returned from that particular landmark.

To summarise, the measurement vector for the Doppler-Azimuth radar at discrete-time \( k \) (\( Z_k \)) is modelled as follows:

For each landmark \( j \), the landmark is either “hit”, or “missed”. The landmark is hit with probability \( P_d(m^{(j)}) \) from Equation (4.7). If the landmark is hit, the Doppler shift \( d_k^{(j)} \), and azimuth \( \phi_k^{(j)} \) (from Equations (3.12) and (3.13), respectively) are measured and corrupted by Gaussian noise:

\[
z_k^{(j)} = [d_k^{(j)}, \phi_k^{(j)}]^T + p_k
\]

Where \( p_k \sim \mathcal{N}(0_{2 \times 1}, r_k) \) with covariance: \( r_k = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix} \)

The probability that the landmark has been missed is therefore \( 1 - P_d(m^{(j)}) \). If the landmark has been missed, an empty set is returned for landmark \( j \): \( z_k^{(j)} = [\emptyset, \emptyset]^T \) which will not increase the cardinality, \( L_k \), of the measurement set.

Furthermore, the measurement set can also be corrupted by false detections which are spurious measurements (clutter) which increases the cardinality (\( L_k \)) of the measurement set. The number of false detections in the measurement set, at discrete-time \( k \), is \( \rho_k \). \( \rho_k \) is distributed according to a Poisson distribution with mean \( \lambda \). If a false detection has occurred, i.e. \( \rho_k > 0 \), then for each of the false detections, the Doppler and azimuth measurements are drawn from uniform distributions:

\[
f_k^{(i)} \sim [\mathcal{U}(-D,D), \mathcal{U}(0,2\pi)]^T
\]

where \( i = 1, 2, \ldots, \rho_k \). The parameter \( D \) should be chosen to include the possible range of Doppler values that could be obtained in the scenario.
This measurement model is highly non-linear, non-Gaussian as even the cardinality of the measurement set is stochastic. Furthermore, there is no prior information known to the particle filter as to which measurement is associated with which landmark, if any (consider the possibility that all landmarks have been missed and all measurements are false detections).

### 4.3.1 Association Hypotheses

Since the associations between landmarks and measurements are unknown, in addition to the possibility of false, and missed detections, the RFS PF must consider all possible association hypotheses (except for the hypothesis where all landmarks are missed by the radar) to compute the weight of each particle.

\( \alpha_h(j) \) is the association function from the list of indices of landmarks \( (j = 1, 2, \cdots, M) \) to the list of indices of measurements in \( Z_k (0, 1, 2, \cdots, L_k) \) with an additional index of 0 for a missed detection i.e. \( \alpha_h(j) : 1, 2, \cdots, M \to 0, 1, 2, \cdots, L_k \). Explained differently, given the landmark \( j \), \( \alpha_h(j) \) will return one of the measurement numbers (or 0, representing a missed detection): \( \alpha_h(j) \in 0, 1, 2, \cdots, L_k \). \( h \) represents the hypothesis number and is a reminder that the association function is required for each hypothesis, for each landmark. This relationship is illustrated in Table 4.1.

<table>
<thead>
<tr>
<th>Hypothesis 1</th>
<th>Landmark 1</th>
<th>Landmark 2</th>
<th>...</th>
<th>Landmark M</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha_1(1) )</td>
<td>( \alpha_1(2) )</td>
<td>...</td>
<td>( \alpha_1(M) )</td>
<td></td>
</tr>
<tr>
<td>Hypothesis 2</td>
<td>( \alpha_2(1) )</td>
<td>( \alpha_2(2) )</td>
<td>...</td>
<td>( \alpha_2(M) )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Hypothesis H</td>
<td>( \alpha_H(1) )</td>
<td>( \alpha_H(2) )</td>
<td>...</td>
<td>( \alpha_H(M) )</td>
</tr>
</tbody>
</table>

Table 4.1: Association function in relation to hypotheses and landmarks.

It is also assumed that a single landmark can produce at most, one Doppler-azimuth measurement at discrete-time \( k \).

**Example List of Hypotheses**

To illustrate with a simple example, assume that the known map contains two landmarks \( (M = 2) \) and at discrete-time \( k \), three Doppler-azimuth measurements have been received \( (L_k = 3) \). The full list of hypotheses is shown in Table 4.2.
Hypothesis 1 describes the possibility that both of the known landmarks have been missed (associated with 0) which therefore implies that the three Doppler-azimuth measurements are all false detections. Hypothesis 2 describes the possibility that Landmark 1 has been missed (associated with 0), but Landmark 2 is associated with measurement number 1. This therefore also implies that measurements 2 and 3 are false detections, and so forth.

4.3.2 Likelihood Function

The likelihood can be derived following Mahler [152, Chapter 12]:

\[
\psi(Z_k|\mathbf{x}_k, \mathbf{m}, \mathbf{u}_k) = \kappa(Z_k) \prod_{j=1}^{M} [1 - P_d(m^{(j)})] \cdot \sum_{h=1}^{H} \prod_{i=1, M \text{ s.t. } \alpha_h(i) > 0}^{M} \frac{P_d(m^{(i)}) \cdot g(z^{(\alpha_h(i))}_k|\mathbf{x}_k, \mathbf{m}(i))}{[1 - P_d(m^{(i)})]} \cdot \lambda \cdot c(z^{(\alpha_h(i))}_k) 
\]

(4.8)

Where:

\[
g(z^{(i)}_k|\mathbf{x}_k, \mathbf{m}(j), \mathbf{u}_k) = \mathcal{N}(z^{(i)}_k; h(\mathbf{x}_k, \mathbf{m}(j), \mathbf{u}_k), r_k) 
\]

(4.9)
4.3. MEASUREMENT MODEL

That is, create a Gaussian with covariance $r_k$, mean equal to the predicted measurement given a pose, landmark $j$, and the control signal $(h(x_k, m^{(j)}_k, u_k))$, and then evaluate this Gaussian at the point of the $i^{th}$ received measurement $z^{(i)}_k$.

The variable $\kappa(Z_k)$ in Equation 4.10 is given in [152, Chapter 12]:

$$\kappa(Z_k) = e^{-\lambda \prod_{j=1}^{L_k} \lambda \cdot c(z^{(j)}_k)}$$  \hspace{1cm} (4.10)

It is assumed that the false detections are independent and uniformly distributed in both range and azimuth, therefore:

$$c(z^{(j)}_k) = \mathcal{U}(d^{(j)}_k; -D, D) \cdot \mathcal{U}(\phi^{(j)}_k, 0, 2\pi)$$  \hspace{1cm} (4.11)

Example Likelihood Function

Referring to the scenario described in Table 4.2, the likelihood function in this case, is:

$$\psi(Z_k | x_k, m, u_k) = \kappa(Z_k) \prod_{j=1}^{2} [1 - P_d(m^{(j)})] \cdot \sum_{h=1}^{13} \prod_{i=1,2, \text{s.t. } \alpha_h(i) > 0} P_d(m^{(i)}) \cdot g(z^{(\alpha_h(i))}_k | x_k, m^{(i)}) \cdot \frac{P_d(m^{(i)}) \cdot g(z^{(\alpha_h(i))}_k | x_k, m^{(i)})}{[1 - P_d(m^{(i)})]} \cdot \lambda \cdot c(z^{(\alpha_h(i))}_k)$$

The second product occurs over $i$ landmarks, such that $\alpha_h(i) > 0$ (i.e. when the landmark is not assumed to have been missed), so when the likelihood is expanded:
\[
\begin{align*}
&= \kappa(Z_k) \prod_{j=1}^{2} [1 - P_d(m^{(j)})] \\
&= \prod_{j=1}^{2} \left( \frac{P_d(m^{(2)}) \cdot g(z_k \{\alpha_2(2)\} | x_k, m^{(2)})}{[1 - P_d(m^{(2)})] \cdot \lambda \cdot c(z_k \{\alpha_2(2)\})} + \frac{P_d(m^{(2)}) \cdot g(z_k \{\alpha_3(2)\} | x_k, m^{(2)})}{[1 - P_d(m^{(2)})] \cdot \lambda \cdot c(z_k \{\alpha_3(2)\})} \right) \\
&\quad \times \left( \frac{P_d(m^{(1)}) \cdot g(z_k \{\alpha_5(1)\} | x_k, m^{(1)})}{[1 - P_d(m^{(1)})] \cdot \lambda \cdot c(z_k \{\alpha_5(1)\})} \right) + \ldots \right)
\end{align*}
\]

Taking care to exclude the cases where \(\alpha_h(i) = 0\) from the second product. Notice that hypothesis 1 \((h = 1)\), there are no cases where \(\alpha_1(i) > 0\) so it is not part of the likelihood.

### 4.3.3 Murty’s Algorithm

The list of possible association hypotheses grows exponentially with the number of landmarks. It is therefore practically necessary to restrict the number of hypotheses to be processed and this can be done using the Murty’s \(k\)-best assignment algorithm [153]. In this thesis, “\(k\)-best” will be substituted with “\(N_{\text{murty}}\)-best”. First, a cost matrix must be defined:
4.3. MEASUREMENT MODEL

\[
\begin{bmatrix}
c_{(1,1)} & c_{(1,2)} & \cdots & c_{(1,L_k)} & d_{(1,1)} & d_{(1,2)} & \cdots & d_{(1,M)} \\
c_{(2,1)} & c_{(2,2)} & \cdots & c_{(2,L_k)} & d_{(2,1)} & d_{(2,2)} & \cdots & d_{(2,M)} \\
\vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\
c_{(M,1)} & c_{(M,2)} & \cdots & c_{(M,L_k)} & d_{(M,1)} & d_{(M,2)} & \cdots & d_{(M,M)}
\end{bmatrix}
\]

Where:

\[
c_{(i,j)} = -\log(g(z_k^{(j)}|x_k, m^{(i)}, u_k)) - \log(P_d(m^{(i)})) + \log(\lambda \cdot c(z_k^{(j)}))
\]

c_{(i,j)} represents the cost of associating landmark \( m^{(i)} \) to measurement \( z_k^{(j)} \) and is computed calculating the negative logarithm of Equation 4.8. To calculate Equation 4.8, this requires the definitions in Equations 4.9, 4.7 and 4.11 found in Section 4.3.

Furthermore, the cost of declaring landmark \( m^{(i)} \) as undetected is:

\[
d_{(i,j)} = \begin{cases} 
   d_{(i,j)} = -\log(1 - P_d(m^{(i)})), & \text{if } i = j \\
   d_{(i,j)} = \infty, & \text{otherwise}
\end{cases}
\]

Based on the association costs in the cost matrix, Murty’s algorithm will return a list of the \( N_{\text{murty}} \) most-likely/best (cost minimising) association hypotheses [153].

When Murty’s algorithm is applied, the likelihood can be summed across \( N_{\text{murty}} \) hypotheses rather than \( H \) total hypotheses which will save computation time.

For completeness, the likelihood then becomes:

\[
\psi(Z_k|x_k, m, u_k) = \kappa(Z_k) \prod_{j=1}^{M} [1 - P_d(m^{(j)})] \cdot \sum_{h=1}^{N_{\text{murty}}} \prod_{i=1}^{M} \frac{P_d(m^{(i)}) \cdot g(z_k^{(\alpha_h(i))}|x_k, m^{(i)})}{[1 - P_d(m^{(i)})] \cdot \lambda \cdot c(z_k^{(\alpha_h(i))})}
\]

A special case of the proposed algorithm is obtained when only the best association hypothesis is used (i.e. \( N_{\text{murty}} = 1 \)). This case is known as global nearest neighbour (GNN) data association. This method is well-known in target tracking literature [154].
4.4 RFS Particle Filter with KLD-Sampling

The proposed solution involves the use of a random finite set particle filter with KLD-sampling to adaptively select the number of particles to save computational cost, as well as the use of Murty’s algorithm to evaluate only the most-likely hypotheses also for the purpose of reducing computational cost.

4.4.1 KLD-Sampling

Particle filters can be made more computationally efficient by adjusting the number of particles at each discrete-time $k$. In the context of mobile robot localisation, initially, when the uncertainty of the robot’s initial pose is high, it is typical that the state space is liberally covered by a large number of particles to attempt to cover the initial pose.

However, as the particle filter converges to the actual pose, the uncertainty of this estimation should be much smaller than compared to the initial localisation problem. As a result of this, the number of particles can be reduced and still maintain a high estimation accuracy. KLD-sampling is a solution to this problem developed by Fox [122, 155].

KLD-sampling is based on the Kullback-Leibler distance between the true posterior distribution, and the sample-based maximum likelihood estimate (MLE) of the true posterior distribution. The KL-distance is accepted as a standard measure for the difference between probability distributions [122].

KLD-sampling calculates the number of particles required ($N_{KLD}$) to guarantee with probability $1-\delta$, that the Kullback-Leibler distance between the true posterior and the MLE of the true posterior is less than $\epsilon$. The number is calculated:

$$N_{KLD} = \frac{1}{2\epsilon} \chi^2_{c-1,1-\delta}$$  \hspace{1cm} (4.13)

Where $c$ is the count of bins with support (i.e. bins occupied by one or more particles). This therefore requires that the map be divided into bins. For the localisation problem, the bins in the map should be defined along the three dimensions: position $x$, $y$ and heading $\theta$. The number of bins with support is determined during the process of sampling from the motion model. A minimum and maximum number of particles is set by the user-defined parameters: $N_{\text{min}}$ and $N_{\text{max}}$, respectively.
Rather than compute the quantiles of the chi-square distribution in Equation 4.13, the Wilson-Hilferty transformation is applied \([156]\):

\[
N_{KLD} \approx \frac{c - 1}{2\epsilon} \left\{ 1 - \frac{2}{9(c - 1)} + \sqrt{\frac{2}{9(c - 1)} \cdot z_{1-\delta}} \right\}^3
\]  

(4.14)

Here \(z_{1-\delta}\) is the upper \(1 - \delta\) quantile of the standard normal distribution \([122]\). The implementation relies on the predictive belief as an estimate for the posterior distribution \([122\text{, pp. 994-995}]\).
KLD-sampling MCL is presented in Algorithm 3, as described by Fox [122] and Thrun et al. [5, pp. 263-267]. A description of the critical steps in Algorithm 3 are as follows:

1. At each discrete-time index $k$, reset the number of particles drawn ($i$), number of occupied bins ($c$) and the required number of particles $N_{\text{KLD}}$. This occurs on Line 2.

2. Divide the map into bins in the $x$, $y$, and $\theta$ dimensions. At each discrete-time index $k$, reset the status of all bins to empty. This occurs on Line 3.

3. Line 4 begins the main while loop for KLD-sampling. This loop repeats until the number of required particles ($N_{\text{KLD}}$) is exceeded.

4. Update $i$, the number of particles drawn on Line 5.

5. On Line 6, draw an index $n$ with replacement, proportional to the previous weights.

6. On Line 7, apply the motion model to sample particle $i$, $(x_k^i)$, from the probabilistic motion model contingent upon the previous particle from the drawn index $n$ $(x_{k-1}^n)$ and control input $(u_k)$.

7. Calculate the unnormalised weight of particle $i$ based on the measurement likelihood ($\tilde{w}_k = \psi(Z_k|x_k^i, m)$) on Line 8.

8. Check if particle $i$ falls into an empty bin or an occupied bin on Line 9. If the particle falls into a bin that is empty, change the status of the bin to occupied on Line 10. Keep track of the number of bins with support with the counter $c$ on Line 11.

9. On Line 13, recalculate $N_{\text{KLD}}$ according to Equation 4.14 taking into account the new number of occupied bins. This value is constrained by a minimum and maximum number of particles: $N_{\text{min}}$ and $N_{\text{max}}$, respectively.

10. Update the number of particles $N_k$ on Line 17. Normalise the weights on Line 19.
4.4. RFS PARTICLE FILTER WITH KLD-SAMPLING

Algorithm 3 KLD Localisation

1: Input: $\{x_{k-1}^{[i]}, w_{k-1}^{[i]}\}_{i=1}^{N_{k-1}} u_k, Z_k$
2: $i = 0; c = 0; N_{\text{KLD}} = N_{\text{min}}$
3: Form a bin array in the state-space, set all bins to empty
4: while $i \leq N_{\text{KLD}}$ do
5: $i = i + 1$ \Comment{Number of particles drawn}
6: Draw $n$ with probability $\propto w_{k-1}^{[n]}$
7: $x_k^{[i]} \sim \pi(x_k|x_k^{[n]}, u_k)$
8: $\tilde{w}_k^{[i]} = \psi(Z_k|x_k^{[i]}, m)$
9: if $x_k^{[i]}$ falls into an empty bin $b$ then
10: Set $b$ to occupied
11: $c = c + 1$ \Comment{Number of bins with support/occupied bins}
12: if $c > 1$ then
13: $N_{\text{KLD}} = \min \left[ N_{\text{max}}, \max \left\{ N_{\text{min}}, \frac{c-1}{2e} \left( 1 - \frac{2}{9(c-1)} + \sqrt{\frac{2}{9(c-1)^2}} \right)^3 \right\} \right]$ \Comment{Update $N_{\text{KLD}}$}
14: end if
15: end if
16: end while
17: $N_k = i$
18: for $i = 1, 2, ..., N_k$ do
19: $w_k^{[i]} = \frac{\tilde{w}_k^{[i]}}{\sum_{p=1}^{N_k} \tilde{w}_k^{[p]}}$ \Comment{Normalise weights}
20: end for
21: Output: $\{x_k^{[i]}, w_k^{[i]}\}_{i=1}^{N_k}$

Similar to Chapter 5.5, the KLD-sampling algorithm was “wrapped” around the random finite set particle filter. This is possible because KLD-sampling assumes that the particles drawn from a distribution will form a multinomial distribution: that is, a set of different bins in the state space (the bins will be formed in the $x, y$ and $\theta$ axes) and each bin has associated, the number of particles in that bin [122]. This assumption is not violated since the proposed RFS particle filter still abides by this framework. Combining KLD-sampling with the RFS particle filter is able to reduce the computational load compared to a standard RFS particle filter since the number of particles is reduced when the particle filter converges.
4.4.2 Implementation of the RFS Particle filter with KLD-Sampling

Initially, the particles are distributed uniformly across the state space to solve the global localisation problem.

The RFS particle filter with KLD-sampling is presented in Algorithm 4 by Guan et al. [7]. The key steps in Algorithm 4 are as follows:

1. At each discrete-time index $k$, reset the number of particles drawn ($i$), number of occupied bins ($c$), and the required number of particles $N_{KLD}$. This occurs on Line 2.

2. Divide the map into bins in the $x$, $y$, and $\theta$ dimensions. At each discrete-time index $k$, reset the status of all bins to $empty$. This occurs on Line 3.

3. Line 4 begins the main $while$ loop for KLD-sampling. This loop repeats until the number of required particles is exceeded.

4. Line 5 counts the numbers of particles drawn.

5. Lines 7 - 9 follows the standard particle filter procedure of sampling a particle from the proposal distribution $\pi(\cdot)$.

6. Form the cost matrix on Line 10. On Line 11 apply Murty’s algorithm to identify the $N_{murty}$ best hypotheses to be evaluated in the likelihood. Evaluate the likelihood for only the best $N_{murty}$ hypotheses on Line 12 as per Equation 4.12.

7. Lines 14-20 checks if the particle has been drawn into a bin already occupied or not. If the particle falls into a bin that has not previously been occupied, the bin’s status is now set to $occupied$, the count of occupied bins is increased and the number of required particles, $N_{KLD}$ is updated.

Algorithm 4 Random Finite Set Particle Filter with KLD-Sampling

1: **Input:** \( \{ x_{k-1}^{[i]}, w_{k-1}^{[i]} \}_{i=1}^{N_{k-1}} u_k, Z_k \)
2: \( i = 0; \ c = 0; \ N_{KLD} = N_{min} \)
3: Form a bin array in the state-space, set all bins to empty
4: \textbf{while} \( i \leq N_{KLD} \) \textbf{do}
5: \( i = i + 1 \) \hspace{1cm} \( \triangleright \) Number of particles drawn
6: Draw \( n \) with probability \( \propto w_{k-1}^{[n]} \)
7: \( B_k^{[n]} = \frac{\partial f(x_{k-1}, u_k)}{\partial u_k} \bigg|_{x_{k-1}^{[n]}, u_k} \)
8: \( Q_k^{[n]} \approx B_k^{[n]} D_k B_k^{[n]T} \)
9: \( x_k^{[i]} \sim \pi(x_k | x_{k-1}^{[i]}, u_k) \) from Equation 3.8
10: Form the cost matrix using \( x_k^{[i]}, m, \) and \( Z_k \)
11: Run Murty’s algorithm to find \( N_{\text{murty}} \) best assignments
12: Compute \( \psi(Z_k | x_k^{[i]}, m, u_k) \) using \( N_{\text{murty}} \) assignments from Equation 4.12
13: \( \tilde{w}_k^{[i]} = \psi(Z_k | x_k^{[i]}, m, u_k) \)
14: \textbf{if} \( x_k^{[i]} \) falls into an empty bin \( b \) \textbf{then}
15: \hspace{1cm} Set \( b \) to occupied
16: \hspace{1cm} \( c = c + 1 \) \hspace{1cm} \( \triangleright \) Number of bins with support/occupied bins
17: \textbf{if} \( c > 1 \) \textbf{then}
18: \hspace{1cm} \( N_{KLD} = \min \left[ N_{\text{max}}, \max \left\{ N_{\text{min}}, \frac{1 - \frac{2}{9(c-1)} + \sqrt{\frac{2}{9(c-1)} z_1 - \delta}}{2c} \right\} \right] \)
19: \textbf{end if}
20: \textbf{end if}
21: \textbf{end while}
22: \( N_k = i \)
23: \textbf{for} \( i = 1, 2, ..., N_k \) \textbf{do}
24: \hspace{1cm} \( w_k^{[i]} = \frac{\tilde{w}_k^{[i]}}{\sum_{p=1}^{N_k} \tilde{w}_k^{[p]}} \) \hspace{1cm} \( \triangleright \) Normalise weights
25: \textbf{end for}
26: \textbf{Output:} \( \{ x_k^{[i]}, w_k^{[i]} \}_{i=1}^{N_k} \)

4.4.3 Simulation Results

The top-down view of the simulated scenario where the algorithm was applied is shown in Figure 4.1. The known map covers an area of 12m \( \times \) 12m and
CHAPTER 4. LOCALISATION USING DOPPLER RADAR: ADVANCED

contained $M = 7$ landmarks. The seven landmarks are positioned:

$$m = \begin{bmatrix} -7 & 4 & -5 & 1 & 3 & -6 & 6 \\ 7 & -5 & -5 & 4 & 3 & 0 & -3 \end{bmatrix}^T$$

The map was divided into bins with the bin size in $x$, $y$ and $\theta$ equal to 0.5 m, 0.5 m and $\frac{10\pi}{180}$ radians, respectively.

The duration of one trial was 30 seconds, with the sampling interval $\Delta t = 1$ second. The initial ground truth for every Monte Carlo trial, was drawn from a Gaussian distribution:

$$x_0 \sim N\left(\begin{bmatrix} -4, -2, \frac{-70\pi}{180} \end{bmatrix}^T, P_0\right)$$

with the initial covariance $P_0 = \text{diag}\left(\sigma_d^2, \sigma_d^2, \left(\frac{40\pi}{180}\right)^2\right)$. The trajectory from the velocity motion model begins at $\begin{bmatrix} -4, -2, \frac{-70\pi}{180}\end{bmatrix}^T$ and is created with a constant control vector: $u_k = [0.3, \frac{6\pi}{180}]^T$, for $k = 1, 2, \cdots, 30$. For the matrix $D_k$ in Equation 3.5, the following parameters were selected: $\gamma_1 = 0.0125$, $\gamma_2 = 0.0125$, $\gamma_3 = 0.05$, and $\gamma_4 = 0.05$. The measurement parameters were $f_c = 10.525$ GHz, noise covariance $r_k = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix}$, with $\sigma_d = 5$ Hz, and $\sigma_\phi = \frac{9\pi}{180}$ radians.

The number of false detections occurred with an average rate of $\lambda$, which varied across different scenarios. The spatial distribution of each false detection, $c(z_k^{(j)})$, was uniform. False Doppler detections were distributed $\mathcal{U}(-D, D)$ with $D = 45$ Hz. The false azimuth detections were distributed $\mathcal{U}(0, 2\pi)$ radians.

The parameter $\beta$ for the probability of detection in Equation 4.7 was set as $\beta = 0.01$.

The RFS particle filter was initialised with $N_0 = 100,000$ particles. The Cartesian coordinates of the particles were distributed uniformly within the map area and the headings were distributed $\mathcal{U}(0, 2\pi)$. The maximum number of particles was set at the initial number $N_{\text{max}} = N_0$, the minimum number of particles $N_{\text{min}} = 25$ with the KLD-sampling parameters $\epsilon = 0.05$ and $\delta = 0.05$.

Figure 4.1 shows a single run of the RFS PF with KLD-sampling. The parameters used were $\lambda = 2$ and $N_{\text{murity}} = 5$. The trajectories are displayed as follows: ground truth (blue line), velocity motion model estimate (red line)
and the weighted mean of the RFS PF with KLD-sampling (green line). In this instance, the particle filter successfully converges to the ground truth.

Figure 4.1: A single realisation of the RFS particle filter.

Monte Carlo simulation results were obtained by averaging over $T = 300$ trials, less the number of failures. The false detection rate $\lambda$ varies with the first letter: A, B, and C, corresponding to $\lambda = 2$, $\lambda = 5$ and $\lambda = 10$, respectively. The following number 10, 5 and 1 correspond to the number of Murty hypotheses $N_{\text{murty}} = 10$, 5 and 1, respectively. $X$ represents the case where the true measurement to landmark associations are known to the particle filter and represents a benchmark. The case where $N_{\text{murty}} = 1$ corresponds to the global nearest neighbour (GNN) algorithm.

The second column of Table 4.3 displays the root mean square error (RMSE) in the robot’s position at the final discrete-time index ($k = 30$). The RMSE in position at any arbitrary discrete-time $k$ is computed as:
\[ \varepsilon_{xy}(k) = \sqrt{\frac{\sum_{\text{trial}=1}^{T}(\hat{x}^{(\text{trial})}_k - x^{(\text{trial})}_k)^2 + \sum_{\text{trial}=1}^{T}(\hat{y}^{(\text{trial})}_k - y^{(\text{trial})}_k)^2}{T}} \]

The cases where the particle filter estimation failed were removed from the calculation. Hence, \( \bar{T} = T - \text{Number of Failures} \). A failure of the particle filter is declared if any of the final ten estimates from the particle filter have an RMSE exceeding a distance of 1.5 meters from the true pose.

RMSE for the heading, \( \varepsilon_{\theta}(k) \), is calculated identically to Equation 3.21. To mitigate angle wrapping issues in calculating the RMSE for the \( \theta \)-axis, an ad hoc process was followed: the heading of each particle was wrapped from \([-\pi, \pi]\) and then wrapped from \([0, 2\pi]\). The angle wrapping scheme that produced the minimum squared error to the true heading was individually chosen for each particle. Then, \( \theta_k^{(\text{trial})} \) as well as \( \hat{\theta}_k^{(\text{trial})} \), defined in Equation 2.11 for each trial, at every discrete-time index \( k \) were each wrapped from \([-\pi, \pi]\) and then wrapped from \([0, 2\pi]\). Of the four possible outcomes, the wrapping scheme that minimised the squared error between the two variables was used to report the RMSE. Similarly, the heading from the velocity motion model estimate for every trial, and \( \theta_k^{(\text{trial})} \) were each wrapped from \([-\pi, \pi]\) and then wrapped from \([0, 2\pi]\). Of the four possible outcomes, the wrapping scheme that minimised the squared error between the two variables was used to report the RMSE.

The fourth column of Table 4.3 the number of particles averaged over the 30 seconds (time averaged) for each of the \( \bar{T} \) trials, and again averaged over \( \bar{T} \) trials (ensemble averaged).
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Final $\varepsilon_{xy}$ (m)</th>
<th>Final $\varepsilon_{\theta}$ (rad)</th>
<th>Time and Ensemble-Averaged Number of Particles</th>
<th>Simulation Time per Trial (sec)</th>
<th>Probability of Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>AX</td>
<td>0.2824</td>
<td>0.0622</td>
<td>7025</td>
<td>39</td>
<td>0.000</td>
</tr>
<tr>
<td>A10</td>
<td>0.2932</td>
<td>0.0724</td>
<td>11002</td>
<td>701</td>
<td>0.060</td>
</tr>
<tr>
<td>A5</td>
<td>0.3000</td>
<td>0.0770</td>
<td>10999</td>
<td>599</td>
<td>0.053</td>
</tr>
<tr>
<td>A1</td>
<td>0.3305</td>
<td>0.0794</td>
<td>11175</td>
<td>490</td>
<td>0.060</td>
</tr>
<tr>
<td>BX</td>
<td>0.2842</td>
<td>0.0577</td>
<td>7004</td>
<td>44</td>
<td>0.003</td>
</tr>
<tr>
<td>B10</td>
<td>0.3135</td>
<td>0.0741</td>
<td>12905</td>
<td>1439</td>
<td>0.070</td>
</tr>
<tr>
<td>B5</td>
<td>0.3290</td>
<td>0.0764</td>
<td>12843</td>
<td>971</td>
<td>0.103</td>
</tr>
<tr>
<td>B1</td>
<td>0.3327</td>
<td>0.0787</td>
<td>12748</td>
<td>830</td>
<td>0.100</td>
</tr>
<tr>
<td>CX</td>
<td>0.2791</td>
<td>0.0605</td>
<td>7016</td>
<td>53</td>
<td>0.007</td>
</tr>
<tr>
<td>C10</td>
<td>0.3549</td>
<td>0.0848</td>
<td>15958</td>
<td>1958</td>
<td>0.113</td>
</tr>
<tr>
<td>C5</td>
<td>0.3492</td>
<td>0.0823</td>
<td>16182</td>
<td>2102</td>
<td>0.140</td>
</tr>
<tr>
<td>C1</td>
<td>0.3651</td>
<td>0.0908</td>
<td>16405</td>
<td>2501</td>
<td>0.180</td>
</tr>
</tbody>
</table>

Table 4.3: RFS PF simulation results under various assumptions.
Table 4.3 shows that in the scenarios when the landmark-measurement associations are known (Scenarios AX, BX and CX), the estimation accuracy is the best. This is expected since having the known associations eliminates any uncertainty from the estimated assignments which are potentially incorrect. Increasing $N_{\text{murty}}$ in Scenarios A and B, leads to improved estimation accuracy when measured by RMSE. In Scenario C, when the false detection rate $\lambda$ is increased to 10, increasing $N_{\text{murty}}$ decreases the probability of failure but does not seem to improve the estimation accuracy when measured by RMSE. The global nearest neighbour solution, when $N_{\text{murty}} = 1$, is never the best estimation by any metric, suggesting that there are benefits to estimation accuracy by evaluating multiple hypotheses.

Table 4.3 shows that Scenarios AX, BX and CX where the landmark-measurement associations are known, have the lowest computational time, by far. This is because Murty’s algorithm is not required. Beyond this, it appears that increasing the false detection rate, $\lambda$, from 2 to 5 to 10, (A to B to C) which would increase the uncertainty of the scenario, leads to an increasing number of particles using KLD-sampling, which is expected to increase computation time.

The Monte Carlo results for Scenario C are presented. Figure 4.2a displays the RMSE in position (meters) vs. time. Figure 4.2b displays the RMSE in heading (radians) vs. time and Figure 4.2c displays the average number of particles vs. time.
4.4. RFS PARTICLE FILTER WITH KLD-SAMPLING

(a) Error in distance (meters) for scenario C.

(b) Error in heading (radians) for scenario C.

(c) Number of particles for scenario C.

Figure 4.2: Monte Carlo results for scenario C.
The results also prove that the KLD-sampling algorithm can be applied to problems involving more complicated measurement models and the corresponding likelihood calculations.

4.5 Conclusion

This chapter reduces the limiting assumptions from the previous chapter on applying the Doppler radar for mobile robot localisation in a GPS-denied environment that is feature-based. The Doppler radar is now assumed to be able to miss the landmark and also produce false detections. The landmark-measurement associations are also unknown. This chapter presents the random finite set (RFS) particle filter as the solution to this complex problem.

This chapter also addresses issues in computation by combining the RFS particle filter with Murty’s algorithm so that only the most likely hypotheses are being evaluated rather than the entire list of hypotheses which will grow exponentially with the number of landmarks in the map. Furthermore, the RFS particle filter is also combined with the KLD-sampling algorithm which will reduce the number of particles as the particle filter converges which further eases computational requirements.

The simulation assumptions for this chapter are more realistic and the results show that the use of a Doppler-azimuth radar as an exteroceptive sensor for localisation is a promising idea when combined with the RFS particle filter. The results from the simulation demonstrate that there are benefits to estimation accuracy when evaluating multiple hypothesis when using Murty’s algorithm. The number of particles increase, along with computation time, when the false detection rate increases.
Chapter 5

KLD-Gmapping

5.1 Introduction

This chapter introduces the influential “Gmapping” proposal distribution. This proposal attempts to draw particles from regions in the state space where the likelihood is very high. It does so by means of numerical optimisation to identify regions of the state space corresponding to the maximum likelihood (ML). Previously, feature-based maps were studied, but this chapter presents localisation on a grid-based map which is better for capturing more detail in dense environments such as inside an office.

The Gmapping proposal discussed in Section 5.2 is computationally expensive. This chapter successfully combines KLD-sampling (previously discussed in Chapter 4.4.1, which adapts the number of particles in the particle filter) with the intelligent Gmapping proposal distribution, showing that those two well-known algorithms can be combined together to create an algorithm: “KLD-Gmapping”, that inherits useful properties from its constituents. The resulting algorithm is studied in simulation and on a set of experimental data assuming the use of a LIDAR (light detection and ranging) as the exteroceptive sensor.

The remainder of the chapter is outlined as follows. Section 5.4 presents the LIDAR measurement model as a weighted mixture of a number of different probabilistic distributions reflecting different sources of noise. KLD-Gmapping is applied to a simulation and a set of experimental data. The results are presented in Section 5.5.1 and Section 5.5.2, respectively. The results of this chapter were published by Guan et al. to a conference paper.
5.2 Gmapping Proposal Distribution

The Gmapping proposal distribution ("Gmapping") is also another solution to mitigate the particle filter degeneracy problem discussed in Chapter 3.7. Gmapping also incorporates the recent measurement into the prior distribution to make the prior distribution more fitted to the likelihood and hence, a better representation for the posterior distribution.

Unlike the EDH particle flow particle filter discussed in Chapter 3.7, the Gmapping proposal distribution relies on the use of numerical optimisation.

This numerical optimisation has the disadvantage of having a longer computation time compared to the elegant closed-form solutions used in the particle flow particle filter. However, it is very useful for complicated likelihoods where a closed-form solution cannot be derived.

Referring to Algorithm 5, Gmapping has several steps in its implementation, as described by Grisetti et al. 105.

1. Line 2 of Algorithm 5 is the main for loop for all particles with index $i = 1, 2, \ldots, N$.

2. Sample a pose from the transitional density, $\pi(\cdot)$:

\[
\tilde{x}_k[i] \sim \pi(x_k|x_{k-1}^{[i]}, u_k)
\]

This occurs on Line 3 of Algorithm 5.

This transitional density could be based on velocity control input signals used in the velocity motion model as per Chapter 3.2.1 or on an inertial-based motion model in Chapter 5.3.

3. Within a limited vicinity of $\tilde{x}_k[i]$, find the pose that maximises the likelihood function ($\psi(\cdot)$) of the current measurement:

\[
\hat{x}_k[i] = \arg \max_{x_k} \psi(Z_k|x_k, m)
\]

The likelihood function depends on the type of exteroceptive sensor and its modelling assumptions. For example, Chapter 3.3 assumes the use
of a Doppler radar with basic noise assumptions. Chapter 4.3 assumes the use of a Doppler radar with more realistic noise assumptions and Chapter 5.4 assumes the use of a LIDAR. Each of these scenarios will result in a different likelihood function.

The optimisation process is initialised at $\hat{x}_k^{[i]}$. This thesis used MATLAB’s fmincon function to perform the optimisation. The optimisation must be constrained by a parameter: $\alpha = \ell \cdot [\sigma_x, \sigma_y, \sigma_\theta]^T$, where $\ell$ is a user-defined parameter. The constraints should be a function of the standard deviation of the proposal distribution because a wider proposal distribution should allow for a wider optimisation window where the maximum likelihood (ML) pose can occur, and vice versa. This step is performed on Line 4.

4. Uniformly sample $U$ particles within a limited vicinity of $\hat{x}_k^{[i]}$, the maximum-likelihood pose:

$$x_a \sim U(\hat{x}_k^{[i]} - \Delta, \hat{x}_k^{[i]} + \Delta)$$  \hspace{1cm} (5.2)

Where the index of the uniformly sampled particles: $a = 1, 2, \ldots, U$. The constant $\Delta$ is a user-defined vector to limit the vicinity of the uniform sampling to be near the maximum likelihood pose: $\Delta = [\Delta_x, \Delta_y, \Delta_\theta]^T$. This occurs on Line 6.

5. Construct a Gaussian distribution based on the statistics of the uniform samples. This is the Gmapping proposal distribution. The Gaussian distribution has mean $\mu_k^{[i]}$ and covariance $\Sigma_k^{[i]}$ computed as follows:

(a) $\mu_k^{[i]} = \frac{1}{\eta^{[i]}} \sum_{a=1}^{U} x_a \cdot \psi(Z_k|x_a, m) \cdot \pi(x_a|x_{k-1}, u_k)$

(b) $\Sigma_k^{[i]} = \frac{1}{\eta^{[i]}} \sum_{a=1}^{U} \psi(Z_k|x_a, m) \cdot \pi(x_a|x_{k-1}, u_k) \cdot (x_a - \mu_k^{[i]}) \cdot (x_a - \mu_k^{[i]})^T$

(c) $\eta^{[i]}$ is a normalisation factor: $\eta^{[i]} = \sum_{a=1}^{U} \psi(Z_k|x_a, m) \cdot \pi(x_a|x_{k-1}, u_k)$

$\mu_k^{[i]}$ and $\eta^{[i]}$ are initialised on Line 8. $\Sigma_k^{[i]}$ is initialised on Line 14. The computation of $\mu_k^{[i]}$ occurs on Line 10 and it is later normalised on Line 13. The computation of $\eta^{[i]}$ occurs on Line 11. $\Sigma_k^{[i]}$ is calculated on Line 16 and is later normalised on Line 18.
6. Sample a pose from the Gaussian Gmapping proposal distribution:

\[ x_k^{[i]} \sim \mathcal{N}(\mu_k^{[i]}, \Sigma_k^{[i]}) \]

This occurs on Line 19.

7. Update the importance weight of particle \( i \), as per \[105\], Equation 18, Algorithm 1:

\[
\tilde{w}_k^{[i]} \approx w_{k-1}^{[i]} \sum_{a=1}^{U} \psi(Z_k | x_a, m) \cdot \pi(x_a | x_{k-1}, u_k) = w_{k-1}^{[i]} \cdot \eta^{[i]}
\]

The importance weight is calculated on Line 20.

8. On Line 23, normalise the weights so that they sum to one: \( w_k^{[i]} = \frac{w_k^{[i]}}{\sum_{p=1}^{N} w_p^{[p]}} \), for \( i = 1, \cdots, N \).

9. Calculate the effective sample size (ESS) based on the normalised weights. A common approximation for the ESS is:

\[
\text{ESS} \approx \left( \sum_{i=1}^{N} \left( w_k^{[i]} \right)^2 \right)^{-1}
\]

[93] p. 3946], [92] pp. 386-387] as per Equation 2.8. If it is less than a user-defined threshold \( (N_{\text{threshold}}) \), resample the particles. Following a resample, the weights are reinitialised uniformly as \( w_k^{[i]} = \frac{1}{N} \), for \( i = 1, \cdots, N \). This is explained in more detail in Chapter 2.5.2. The ESS is calculated on Line 25. If the ESS is below the threshold, resample on Line 27.

10. Proceed to the next time sample and repeat.

Algorithm 5 presents the Gmapping proposal distribution.
Algorithm 5 Gmapping Proposal Distribution

1: Input: \( \{ x_{k-1}^i, w_{k-1}^i \}_{i=1}^N u_k, Z_k \)
2: for \( i = 1 : N \) do
3: \( \hat{x}_k^i \sim \pi(x_k|x_{k-1}^i, u_k) \)
4: \( \bar{x}_k^i = \arg\max_{x_k} \psi(Z_k|x_k, m) \), subject to \( \bar{x}_k^i \in \{ \bar{x}_k^i - \alpha, \bar{x}_k^i + \alpha \} \)
   \( \triangleright \) Find the pose that maximises the likelihood.
5: for \( a = 1 : U \) do
6: \( x_a \sim U(\bar{x}_k^i - \Delta, \bar{x}_k^i + \Delta) \)
   \( \triangleright \) Sample \( U \) particles uniformly around the maximum likelihood pose.
7: end for
8: \( \mu_k^i = [0, 0, 0]^T \); \( \eta^i = 0 \)
9: for \( a = 1 : U \) do
10: \( \mu_k^i = \mu_k^i + x_a \cdot \psi(Z_k|x_a, m) \cdot \pi(x_a|x_{k-1}^i, u_k) \)
11: \( \eta^i = \eta^i + \psi(Z_k|x_a, m) \cdot \pi(x_a|x_{k-1}^i, u_k) \)
12: end for
13: \( \mu_k^i = \mu_k^i / \eta^i \)
14: \( \Sigma_k^i = 0_{3 \times 3} \)
15: for \( a = 1 : U \) do
16: \( \Sigma_k^i = \Sigma_k^i + \psi(Z_k|x_a, m) \cdot \pi(x_a|x_{k-1}^i, u_k) \cdot (x_a - \mu_k^i) \cdot (x_a - \mu_k^i)^T \)
17: end for
18: \( \Sigma_k^i = \Sigma_k^i / \eta^i \)
19: \( x_k^i \sim \mathcal{N}(\mu_k^i, \Sigma_k^i) \)
   \( \triangleright \) Sample from the newly formed Gmapping proposal distribution.
20: \( \tilde{w}_k^i = w_k^i \cdot \eta^i \)
   \( \triangleright \) Calculate the weight of the particle.
21: end for
22: for \( i = 1, 2, ..., N \) do
23: \( w_k^i = \frac{\tilde{w}_k^i}{\sum_{i=1}^N \tilde{w}_k^i} \)
   \( \triangleright \) Normalise weights
24: end for
25: \( \text{ESS} = 1 / \sum_{i=1}^N (w_k^i)^2 \)
26: if \( \text{ESS} < N_{\text{threshold}} \) then
27: Resample: \( \{ x_k^i, w_k^i \}_{i=1}^N \rightarrow \{ x_k^i, 1/N \}_{i=1}^N \)
   \( \triangleright \) If the effective sample size is below the threshold, resample.
28: end if
29: Output: \( \{ x_k^i, w_k^i \}_{i=1}^N \)
5.3 Inertial Navigation-Based Motion Model

The motion model used in Chapter 3.2.1 assumes that the translational and rotational velocity are known. If this data is unavailable, an inertial measurement unit (IMU) can also provide a crude estimate of how the robot has moved. This type of motion model is inertial navigation-based. Referring to Equation (3.1), the control input applied from \( k - 1 \) to \( k \) comprises of a change in the \( x, y \) and \( \theta \) axes.

The IMU contains an accelerometer and a gyroscope. The data from the accelerometer can be double-integrated over a time period to infer a change in position of the robot. Similarly, the data from the gyroscope can be integrated over a time period to obtain a change in heading. This however can only be performed over short distances as the noise will quickly accumulate.

For the experiment described in Section 5.5.2 an OptiTrack motion capture system was available. This motion capture system provides \( \mathbf{x}_k \), the ground truth robot pose. In this experiment, proprioceptive data was not available. However, inertial navigation data was simulated by taking the data from the OptiTrack motion capture system (which reports the ground truth) and adding biased noise to this data. The difference in pose between samples was then used as the simulated inertial navigation data.

The simulated inertial navigation data \( \mathbf{x}_k^{\text{in}} = [\mathbf{x}_k^{\text{in}}, \mathbf{y}_k^{\text{in}}, \mathbf{\theta}_k^{\text{in}}]^T \) is constructed:

\[
[\mathbf{x}_k^{\text{in}}, \mathbf{y}_k^{\text{in}}, \mathbf{\theta}_k^{\text{in}}]^T = [\mathbf{x}_{k-1}^{\text{in}}, \mathbf{y}_{k-1}^{\text{in}}, \mathbf{\theta}_{k-1}^{\text{in}}]^T + [\Delta x_k, \Delta y_k, \Delta \theta_k]^T + \bar{e}_k 
\]

where:

\[
[\Delta x_k, \Delta y_k, \Delta \theta_k]^T = [x_k, y_k, \theta_k]^T - [x_{k-1}, y_{k-1}, \theta_{k-1}]^T 
\]

obtained by differencing the ground truth robot pose obtained from the OptiTrack motion capture system.

The simulated inertial navigation noise was distributed:

\[
\bar{e}_k \sim \mathcal{N}(\mathbf{B}, \mathbf{Q}) 
\]

\( \mathbf{B} \) is a bias in the noise that cannot be fully modelled. It is used to make the simulated inertial navigation data more realistic. The bias is unknown to the particle filter.

The simulation results from Chapter 5.5.1 used the velocity motion model described in Chapter 3.2.1.
5.4 LIDAR MEASUREMENT MODEL

For the experiment as well as the simulation, the motion noise covariance $Q$ was a constant.

5.4 LIDAR measurement model

A LIDAR is a sensor which detects distance in a (semi) circular pattern. The LIDAR has a total of $D$ beams and the angular spacing between beams is a property of the LIDAR hardware. Following Thrun et al. [5, Chapter 6.3], a measurement along one LIDAR beam $d$, consists of the measured distance to the obstacle, $z_k^{(d)}$. However, it is also necessary to establish a true distance from the robot to an obstacle, referred to as a “raycast”. The raycast is determined by measuring the distance from a pose to the obstacle on the map. The raycast along beam $d$ at sample $k$ is equal to the distance:

$$z_k^{(d)*} = \sqrt{(x_k - x_d)^2 + (y_k - y_d)^2}$$

Where $(x_d, y_d)$ are the coordinates of the nearest object struck by the $d$’th beam. The measurement received from each beam is assumed to be equal to the raycast plus additive noise:

$$z_k^{(d)} = z_k^{(d)*} + n_k^{(d)}$$

The total measurement set at discrete-time index $k$, is the combined set of all $1, 2, \cdots, D$ beams:

$$Z_k = [z_k^{(1)}, z_k^{(2)}, \cdots, z_k^{(D)}]^T$$

Figure 5.1 shows an approximate illustration of the scenario. The environment is modelled by an occupancy grid-based map. The mobile robot is carrying a LIDAR with $1, 2, \cdots, D$ beams which reflect off objects in the environment producing a noisy measurement. However, a few features of this illustration may be subject to change. Beam number 1 is not necessarily originating from the front of the robot, nor is it clear if the beams are incrementing in a clockwise/counter-clockwise direction. This all depends on the orientation and design of the LIDAR sensor. Also, the angular spacing between beams can vary, since in Chapter 5.5.2 a number of beams were discarded from the dataset.
Figure 5.1: Illustration of the KLD-Gmapping scenario.
As proposed in [5], the likelihood function of range measurements for a single beam \(d\), \(\psi(z_k^{(d)}|x_k, m)\) is modelled as a weighted sum of four densities, each representing a different source of noise. The final likelihood would resemble Figure 5.2 or Figure 5.9 depending on the parameters and weightings of the four densities. The four densities are described in the proceeding subsections. The noise for each beam, \(n_k^{(d)}\) is distributed according to the combined weighted density.

### 5.4.1 Laser beam successfully hits an object in the map and returns a measurement

The case where a laser beam successfully hits an object within the LIDAR’s maximum range is modelled as a truncated Gaussian distribution. The mean of the truncated Gaussian distribution is equal to the true distance to the object, which gives:

\[
\psi_{\text{hit}}(z_k^{(d)}|x_k, m) = \eta_{\text{hit}} \cdot \mathcal{N}(z_k^{(d)}; z_k^{(d)*}; \sigma_{\text{hit}}^2), \quad \text{if } 0 \leq z_k^{(d)} \leq z_{\text{max}}
\]  

Unlike a proper Gaussian distribution, \(\psi_{\text{hit}}(z_k^{(d)}|x_k, m)\) is equal to zero if the received measurement for the beam, \(z_k^{(d)}\) is less than zero or greater than the LIDAR’s maximum range \(z_{\text{max}}\). This truncated Gaussian probability density function (PDF) has been truncated to fit within the LIDAR’s range limits. Hence \(\eta_{\text{hit}}\), the normalisation constant is:

\[
\eta_{\text{hit}} = \left(\int_{0}^{z_{\text{max}}} \mathcal{N}(z_k^{(d)}; z_k^{(d)*}; \sigma_{\text{hit}}^2) \, dz_k^{(d)}\right)^{-1}
\]

which normalises the PDF area to 1.

### 5.4.2 Laser beam was blocked by an unexpected object

Truncated exponential noise is used to model the measurement noise occurring in the case when an object, not forming part of the map (e.g. a person) has blocked the beam and produced an unexpectedly short measurement. The probability of this event occurring is higher for closer ranges because unexpected objects far away are less likely to be struck by a LIDAR beam and create a laser return. This is captured in the properties of the truncated exponential distribution:
\( \psi_{\text{short}}(z_k^{(d)} | x_k, m) = \eta_{\text{short}} \cdot \lambda_{\text{short}} \cdot e^{-\lambda_{\text{short}} z_k^{(d)}}, \) if \( 0 \leq z_k^{(d)} \leq z_k^{(d)*} \)

If the received measurement for the beam, \( z_k^{(d)} \) is less than zero or greater than the raycast distance \( z_k^{(d)*} \), \( \psi_{\text{short}}(z_k^{(d)} | x_k, m) \) is equal to zero. Because this pseudo-exponential distribution probability density function has been truncated, the normalisation constant is:

\[
\eta_{\text{short}} = \left( \int_{0}^{z_k^{(d)*}} \lambda_{\text{short}} \cdot e^{-\lambda_{\text{short}} z_k^{(d)}} \, dz_k^{(d)} \right)^{-1} = \frac{1}{1 - e^{-\lambda_{\text{short}} z_k^{(d)*}}}
\]

which normalises the truncated exponential distribution to have an area of 1 within its support: \( 0 \leq z_k^{(d)} \leq z_k^{(d)*} \).

5.4.3 No obstacles were detected within range

Some LIDARs return the value of \( z_{\text{max}} \) when no objects were detected within its maximum range. This can also occur when an emitted laser beam does strike an object within range, but the reflected laser beam is not received. This can happen for example, if the object is glass, or its surface is a good absorber of light, or if the LIDAR beam reflects in the wrong direction, or simply in bright lighting conditions. In any case, the measurement noise model will incorporate a point-mass distribution at \( z_{\text{max}} \) to account for these possibilities:

\[
\psi_{\text{max}}(z_k^{(d)} | x_k, m) = \begin{cases} 
1, & \text{if } z_k^{(d)} = z_{\text{max}} \\
0, & \text{otherwise.}
\end{cases}
\]

5.4.4 Laser returns were random

Finally, an indiscriminate source of noise is added to account for entirely unexplainable, random clutter in the measurements. This noise is uniformly distributed, described by:

\[
\psi_{\text{random}}(z_k^{(d)} | x_k, m) = \begin{cases} 
\frac{1}{z_{\text{max}}}, & \text{if } 0 \leq z_k^{(d)} < z_{\text{max}} \\
0, & \text{otherwise.}
\end{cases}
\]
5.5. **KLD-GMAPPING**

Four weighting coefficients: $w_{\text{hit}}$, $w_{\text{short}}$, $w_{\text{max}}$, and $w_{\text{random}}$ are chosen to reflect the relative likelihood of occurrence of each type of noise. The weighted mixture of these four sources of noise forms the total likelihood of the measurement noise for a single beam:

$$
\psi(z_k^{(d)} | x_k, m) = \begin{bmatrix}
  w_{\text{hit}} \\
  w_{\text{short}} \\
  w_{\text{max}} \\
  w_{\text{random}}
\end{bmatrix}^T \begin{bmatrix}
  \psi_{\text{hit}}(z_k^{(d)} | x_k, m) \\
  \psi_{\text{short}}(z_k^{(d)} | x_k, m) \\
  \psi_{\text{max}}(z_k^{(d)} | x_k, m) \\
  \psi_{\text{random}}(z_k^{(d)} | x_k, m)
\end{bmatrix}
$$

(5.5)

where $w_{\text{hit}} + w_{\text{short}} + w_{\text{max}} + w_{\text{random}} = 1$.

Furthermore, it is assumed that the measurements from each LIDAR beam are independent. Hence, the likelihood of one full set of LIDAR measurements at sample $k$, for all $D$ beams is:

$$
\psi(Z_k | x_k, m) = \prod_{d=1}^{D} \psi(z_k^{(d)} | x_k, m)
$$

(5.6)

### 5.5 KLD-Gmapping

Gmapping per se, assumes a fixed number of particles. It can therefore benefit from having an adaptive number of particles. KLD-sampling typically assumes the use of the motion model as the proposal distribution. For example, Fox [155] seems to rely on odometry (wheel encoder) data without incorporating measurement data for the proposal (transitional) distribution. It can therefore benefit from having a more intelligent proposal distribution. Hence, if it were possible to combine the qualities of these two algorithms, the resulting combination should create a particle filter that is computationally lighter than regular Gmapping, and also higher accuracy than a regular KLD-sampling particle filter.

KLD-sampling assumes that the particles drawn from a distribution will form a multinomial distribution: that is, a set of different bins in the state space (the bins will be formed in the $x$, $y$ and $\theta$ axes) and each bin has associated, the number of particles in that bin [122, p. 993].

Furthermore, in the solution presented by Fox [122, p. 994], the author implements KLD-sampling by “rely(ing) on the sample-based representation of the predictive belief as an estimate for the posterior”.

Based on these assumptions, it is valid to use the Gmapping proposal distribution for the predictive belief (rather than the motion model proposal \( \pi(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \)) suggested in [122], [155] because it serves as an even better estimate for the posterior distribution. Drawing particles from the Gmapping proposal will also create a multinomial distribution in the same way.

Therefore, the KLD-sampling framework was “wrapped” around the Gmapping proposal distribution, resulting in the “KLD-Gmapping” algorithm presented by Guan et al. [8]. The KLD-Gmapping algorithm is similar to the description of Gmapping in Algorithm 5 with the additional KLD-sampling steps similar to Algorithm 3. Referring to KLD-Gmapping in Algorithm 6, the critical steps are:

1. Line 2 initialises the number of particles \( i \), number of occupied bins \( c \) and the required number of particles \( N_{KLD} \).

2. Line 3 requires that the map is divided into bins. Set the status of each bin to empty.

3. Line 4 begins the main while loop for KLD-sampling. This loop exits once the number of required particles is exceeded.

4. Line 5 counts the numbers of particles drawn.

5. Then follows the Gmapping algorithm similar to Algorithm 5.

6. Lines 25-31 checks if the particle has been drawn into a bin that is empty, or not. If the bin is empty, the bin’s status is set to occupied, the count of occupied bins is increased and the number of required particles is updated.

KLD-Gmapping is summarised in Algorithm 6.
Algorithm 6 KLD-Gmapping Algorithm

1: **Input:** $\{x_{k-1}^{[i]}, w_{k-1}^{[i]}\}_{i=1}^{N_{k-1}} u_k, Z_k$
2: $i = 0$; $c = 0$; $N_{KLD} = N_{\text{min}}$
3: Form a bin array in the state-space, set all bins to empty
4: **while** $i \leq N_{KLD}$ **do**
5: $i = i + 1$ \quad // Number of particles drawn
6: Draw $n$ with probability $\propto w_{k-1}^{[n]}$
7: $\tilde{x}_k^{[i]} \sim \pi(x_k|x_{k-1}^{[n]}, u_k)$
8: $\hat{x}_k^{[i]} = \arg\max_{x_k} \psi(Z_k|x_k, m)$, subject to $\hat{x}_k^{[i]} \in \{x_k^{[i]} - \alpha, x_k^{[i]} + \alpha\}$
9: for $a = 1 : U$ **do**
10: $x_a \sim \mathcal{U}(x_k^{[i]} - \Delta, x_k^{[i]} + \Delta)$
11: end for
12: $\mu_k^{[i]} = [0, 0, 0]^T$; $\eta^{[i]} = 0$
13: for $a = 1 : U$ **do**
14: $\mu_k^{[i]} = \mu_k^{[i]} + x_a \cdot \psi(Z_k|x_k, m) \cdot \pi(x_a|x_{k-1}^{[n]}, u_k)$
15: $\eta^{[i]} = \eta^{[i]} + \psi(Z_k|x_k, m) \cdot \pi(x_a|x_{k-1}^{[n]}, u_k)$
16: end for
17: $\mu_k^{[i]} = \mu_k^{[i]} / \eta^{[i]}$
18: $\Sigma_k^{[i]} = 0_{3 \times 3}$
19: for $a = 1 : U$ **do**
20: $\Sigma_k^{[i]} = \Sigma_k^{[i]} + \psi(Z_k|x_k, m) \cdot \pi(x_a|x_{k-1}^{[n]}, u_k) \cdot (x_a - \mu_k^{[i]}) \cdot (x_a - \mu_k^{[i]})^T$
21: end for
22: $\Sigma_k^{[i]} = \Sigma_k^{[i]} / \eta^{[i]}$
23: $x_k^{[i]} \sim \mathcal{N}(\mu_k^{[i]}, \Sigma_k^{[i]})$
24: $\tilde{w}_k^{[i]} = \eta^{[i]}$
25: if $x_k^{[i]}$ falls into an empty bin $b$ **then**
26: Set $b$ to occupied
27: $c = c + 1$ \quad // Number of bins with support/occupied bins
28: if $c > 1$ **then**
29: $N_{KLD} = \min \left[ N_{\text{max}}, \max \left\{ N_{\text{min}}, \frac{c-1}{2\epsilon} \left( 1 - \frac{2}{9(c-1)} + \sqrt{\frac{2}{9(c-1)}} \zeta_1 - \delta \right)^3 \right\} \right]$
30: end if
31: end if
32: end while
33: $N_k = i$
34: for $i = 1, 2, ..., N_k$ **do**
35: $w_k^{[i]} = \frac{\tilde{w}_k^{[i]}}{\sum_{p=1}^{N_k} \tilde{w}_k^{[p]}}$ \quad // Normalise weights
36: end for
37: **Output:** $\{x_k^{[i]}, w_k^{[i]}\}_{i=1}^{N_k}$
5.5.1 KLD-Gmapping Simulation Results

For this simulation based on Guan et al. [8], [10], the velocity motion model described in Chapter 3.2.1 was used. The robot starts at the pose: 

\[ [200 \text{ cm}, 50 \text{ cm}, \frac{\pi}{2} \text{ rad}]^T \] for Scenario One and at 

\[ [800 \text{ cm}, 900 \text{ cm}, \pi \text{ rad}]^T \] for Scenario Two. The robot’s true trajectory was stochastic for each Monte Carlo run due to the random process noise with covariance 

\[ Q_k = \text{diag}(\left[ 6^2, 6^2, \left(\frac{4\pi}{180}\right)^2 \right]), \forall k. \]

For this simulation, the units are centimetres for the \( x \) and \( y \) axes, and radians for the \( \theta \) axis. The sampling interval was \( \Delta t = 1 \) second and each simulation lasted for 20 samples. Both scenarios used the same map that was divided into bins with the bin size in \( x \), \( y \) and \( \theta \) equal to 50 cm, 50 cm and \( \frac{20\pi}{180} \) radians, respectively.

The number of particles to initialise the particle filter is equal to the maximum number of particles \( N_{\text{max}} = 100,000 \). These particles were initially distributed uniformly across the map to solve the global localisation problem. The same set of initial particles were used to start the particle filter for both KLD localisation and KLD-Gmapping algorithms. The following KLD parameters from Equation 4.14 were used for the two algorithms in both, Scenario One and Scenario Two:

<table>
<thead>
<tr>
<th></th>
<th>KLD Localisation</th>
<th>KLD-Gmapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \epsilon )</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>( \delta )</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>( N_{\text{min}} )</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>( N_{\text{max}} )</td>
<td>100000</td>
<td>100000</td>
</tr>
</tbody>
</table>

Table 5.1: KLD parameters for the simulation.

The Gmapping proposal distribution had an optimisation solution constraint determined by: \( \alpha = 1.5 \cdot [\sigma_x, \sigma_y, \sigma_\theta] \) when finding the maximum likelihood pose, as per Equation 5.1. The number of particles distributed uniformly around each maximum likelihood pose was set to \( U = 50 \), distributed in the region defined by: \( \Delta = [5, 5, \frac{4\pi}{180}]^T \), as per Equation 5.2.

When an object is hit by the LIDAR, the noise is distributed with zero mean and variance of \( \sigma_{\text{hit}}^2 = 10^2 \). The rate of the exponential distribution when a short reading is obtained is \( \lambda_{\text{short}} = 0.05 \). The maximum range of the LIDAR is assumed to be \( z_{\text{max}} = 450 \) cm, with \( D = 24 \) beams spread out every \( \frac{15\pi}{180} \) radians to cover a full \( 2\pi \) radians field of view. The weighting of
5.5. **KLD-GMAPPING**

![Example PDF of a single LIDAR beam measurement in simulation.](image)

Figure 5.2: Example PDF of a single LIDAR beam measurement in simulation.

These individual distributions are assumed to be:

\[
\begin{bmatrix}
  w_{\text{hit}} \\
  w_{\text{short}} \\
  w_{\text{max}} \\
  w_{\text{random}}
\end{bmatrix} =
\begin{bmatrix}
  0.928 \\
  0.04 \\
  0.002 \\
  0.03
\end{bmatrix}
\]

An example of the likelihood formed after these weights are applied is shown in Figure 5.2.

A single realisation of the two scenarios is shown in Figure 5.3. In Scenario One, the robot has an initial pose of \( \mathbf{x}_0 = [2 \text{ m}, 0.5 \text{ m}, \pi/2 \text{ rad}]^T \). In Scenario Two, the robot has an initial pose of \( \mathbf{x}_0 = [8 \text{ m}, 9 \text{ m}, \pi \text{ rad}]^T \). The control input signal \( \mathbf{u}_k \) varies with time and was selected to have the robot navigate an acceptable trajectory in the environment.
Figure 5.3: A single realisation of two simulated scenarios comparing KLD localisation against KLD-Gmapping.

After repeating the simulation over $T = 30$ Monte Carlo trials for Scenario One and Scenario Two, the accuracy of KLD localisation compared to KLD-Gmapping was evaluated in Figure 5.4 and Figure 5.5. The number of particles is compared in Figure 5.6 and Figure 5.7. Table 5.2 displays the average computation time (in seconds) required to complete one trial in simulation. The RMSEs were calculated as per Equation 3.21.

To mitigate angle wrapping issues in calculating the RMSE for the $\theta$-axis, an ad hoc process was followed: the heading of each particle was wrapped from $[-\pi, \pi]$ and then wrapped from $[0, 2\pi]$. The angle wrapping scheme that
produced the minimum squared error to the true heading was individually chosen for each particle. Then, $\theta^\text{trial}_k$ as well as $\hat{\theta}^\text{trial}_k$, defined in Equation 2.11 for each trial, at every discrete-time index $k$ were each wrapped from $[-\pi, \pi]$ and then wrapped from $[0, 2\pi]$. Of the four possible outcomes, the wrapping scheme that minimised the squared error between the two variables was used to report the RMSE.
Figure 5.4: Comparison of the RMSEs between KLD localisation and KLD-Gmapping for Scenario One.
Figure 5.5: Comparison of the RMSEs between KLD localisation and KLD-Gmapping for Scenario Two.
Figure 5.6: Comparison of the number of particles between KLD localisation and KLD-Gmapping for Scenario One.

Figure 5.7: Comparison of the number of particles between KLD localisation and KLD-Gmapping for Scenario Two.
Table 5.2: Computation time (in seconds) for KLD localisation and KLD-Gmapping in simulation.

<table>
<thead>
<tr>
<th>Application:</th>
<th>KLD Localisation</th>
<th>KLD-Gmapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation: Scenario One</td>
<td>70</td>
<td>150</td>
</tr>
<tr>
<td>Simulation: Scenario Two</td>
<td>66</td>
<td>127</td>
</tr>
</tbody>
</table>

The results show that KLD-Gmapping provides a more accurate estimation than KLD localisation in the simulations. Although KLD-Gmapping uses fewer particles than KLD localisation, it still requires substantially more computation time. This is because the computational demand is much higher on a per-particle basis when using the Gmapping proposal distribution. The reduction in the number of particles would be useful in mapping applications where each particle stores a stochastic realisation of the map, therefore requiring more memory for each particle. The computations were completed using MATLAB, on a PC with an Intel Core i5 6600 3.30 GHz processor, and 32 GB of RAM. Despite having the same KLD parameters, a larger number of particles for KLD localisation was the result compared to KLD-Gmapping. This is probably because KLD-Gmapping has the particles distributed in a more focused area which requires less particles as there are less bins occupied as per Equation 4.14.

5.5.2 KLD-Gmapping Experiment

The experiment was completed at DST Group’s Fishermans Bend campus, presented by Guan et al. [8]. The experiment lasted for 40 samples and was conducted in an indoor environment with obstacles. The map was divided into bins with the bin size in $x$, $y$, and $\theta$ equal to 0.50 m, 0.50 m, and $\frac{15\pi}{180}$ rad, respectively. The environment was equipped with an OptiTrack motion capture system to track the ground-truth pose of the trolley. A trolley equipped with a Hokuyo LIDAR was used in place of the robot. Conventional velocity control input/odometry/inertial navigation-based sources were not available, so inertial navigation-based data was simulated from the ground-truth with added noise with drift. This is described in Chapter 5.3. The bias for the inertial navigation drift was selected as:

$$B = \begin{bmatrix} 0.012, & -0.012, & \frac{0.6\pi}{180} \end{bmatrix}^T$$
and noise covariance:

$$Q_k = \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_{\theta}^2) = \text{diag} \left( \begin{bmatrix} 0.1^2, 0.1^2, \left( \frac{2\pi}{180} \right)^2 \end{bmatrix} \right), \forall k$$

The LIDAR is assumed to have a maximum range $z_{\text{max}} = 6.4$ m, and has a field of view covering $\pi$ radians. Of the 512 LIDAR measurements received every sample $k$, all measurements with no returning signal were removed, as it was concluded that the majority of these readings occurred due to sensor failure. Furthermore, measurements below 0.5 m were also removed, as they were erroneous readings resulting from obstructions due to the experimental apparatus \[157\]. Of the remaining LIDAR measurements, only 50 measurements were retained and the rest of the measurements were discarded to reduce the computational load.

Figure 5.8 shows a plot of the data received from the LIDAR used in the experiment at sample $k = 10$. The vertical object on the right-hand side of the room (approximately along the $X = 4$ axis) was a net. The net was in the room and was defined as part of the map by the OptiTrack motion capture system. However, this net was not detected by the LIDAR as shown in Figure 5.8 where the measured LIDAR beams pass through the net. This would be a source of error for the particle filter algorithms since the particle filter calculates the likelihood by using the map $m$, which contains the net, evaluated against the measurement data which does not acknowledge the existence of the net. Both particle filters have some limited robustness towards this problem. After the robot has moved beyond the net, KLD localisation should be able to re-establish the robot’s pose as long as there are some particles still within the vicinity of the true pose. This becomes an issue of maintaining particle diversity. In comparison, KLD-Gmapping’s optimisation step based on the LIDAR measurements after the robot has passed the net should help to re-establish the robot’s true pose. The subsequent uniformly sampled particles also play a role in increasing diversity as the particles are spread around the maximum-likelihood point. Again, these means are subject to limitations: the true pose must be within the optimisation limits or within the uniform distribution’s support.

The computations were completed using MATLAB, on a PC with an Intel Core i5 6600 3.30 GHz processor and 32 GB of RAM.
Figure 5.8: Actual measurements received (green lines), at $k = 10$ from the ground-truth trajectory (blue) during the experiment.
The relative probability of receiving a hit, a short reading, a maximum reading and a uniformly distributed, random reading (from Equation [5.5]) have been loosely estimated by viewing plots of the received LIDAR measurements against the true robot pose and true map. The probabilities of short and maximum readings have been set to 0 because these readings were mostly erroneous and were entirely removed from the dataset. The weightings have been chosen as:

\[
\begin{bmatrix}
    w_{\text{hit}} \\
    w_{\text{short}} \\
    w_{\text{max}} \\
    w_{\text{random}}
\end{bmatrix} = \begin{bmatrix}
    0.85 \\
    0 \\
    0 \\
    0.15
\end{bmatrix}
\]

When these weights are applied, this forms a likelihood shown in Figure 5.9.

When an object is hit, as in Equation [5.4], the noise has been loosely estimated to be Gaussian distributed with zero mean and variance \( \sigma_{\text{hit}}^2 = \)
Figure 5.10: A single realisation of KLD-Gmapping compared against the ground truth, KLD localisation and the trajectory from the simulated inertial navigation data.

The number of particles to initialise the particle filter is equal to the maximum number of particles $N_{\text{max}} = 15,000$. These particles were initially distributed uniformly across the map to solve the global localisation problem. The same set of initial particles were used to start the particle filter for both KLD localisation and KLD-Gmapping algorithms. The KLD parameters from Equation 4.14 used for the two algorithms are shown in Table 5.3.
The Gmapping proposal distribution had an optimisation constraint determined by: $\alpha = 3 \cdot [\sigma_x, \sigma_y, \sigma_\theta]$ when finding the ML pose, as per Equation 5.1. The number of particles distributed uniformly around each ML pose was set to $U = 20$, distributed in the region defined by: $\Delta = [0.8, 0.8, \frac{20\pi}{180}]^T$, as per Equation 5.2. Monte Carlo analysis will still be required as there is randomness introduced by the initial uniform distribution of particles, as well as the simulated inertial navigation noise. The RMSEs were calculated as per Equation 3.21 which are restated for convenience:

$$\varepsilon_x(k) = \sqrt{\frac{\sum_{\text{trial}=1}^{T}(\hat{x}_{\text{trial}}(k) - x_{\text{trial}}(k))^2}{T}}$$
$$\varepsilon_y(k) = \sqrt{\frac{\sum_{\text{trial}=1}^{T}(\hat{y}_{\text{trial}}(k) - y_{\text{trial}}(k))^2}{T}}$$
$$\varepsilon_\theta(k) = \sqrt{\frac{\sum_{\text{trial}=1}^{T}(\hat{\theta}_{\text{trial}}(k) - \theta_{\text{trial}}(k))^2}{T}}$$

To mitigate angle wrapping issues in calculating the RMSE for the $\theta$-axis, an ad hoc process was followed: the heading of each particle was wrapped from $[-\pi, \pi]$ and then wrapped from $[0, 2\pi]$. The angle wrapping scheme that produced the minimum squared error to the true heading was individually chosen for each particle. Then, $\hat{\theta}_{\text{trial}}(k)$ as well as $\hat{\theta}_{\text{trial}}(k)$, defined in Equation 2.11 for each trial, at every discrete-time index $k$ were each wrapped from $[-\pi, \pi]$ and then wrapped from $[0, 2\pi]$. Of the four possible outcomes, the wrapping scheme that minimised the squared error between the two variables was used to report the RMSE. Similarly, the heading from the simulated inertial navigation system, $\hat{\theta}_{\text{in}}(k)$ for every trial, and $\hat{\theta}_{\text{trial}}(k)$ were each wrapped from $[-\pi, \pi]$ and then wrapped from $[0, 2\pi]$. Of the four possible outcomes, the wrapping scheme that minimised the squared error between the two variables was used to report the RMSE.

<table>
<thead>
<tr>
<th>KLD Localisation</th>
<th>KLD-Gmapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>0.03</td>
</tr>
<tr>
<td>$\delta$</td>
<td>0.03</td>
</tr>
<tr>
<td>$N_{\text{min}}$</td>
<td>20</td>
</tr>
<tr>
<td>$N_{\text{max}}$</td>
<td>15000</td>
</tr>
</tbody>
</table>

Table 5.3: KLD parameters for the experimental data.
The simulated inertial navigation signal is generously assumed to start at the true initial pose at $[-4.0383 \text{ m}, -0.3057 \text{ m}, -1.4027 \text{ rad}]^T$. The estimation accuracy results of $T = 30$ Monte Carlo trials are shown in Figure 5.11 with the number of particles shown in Figure 5.12. Table 5.4 displays the average computation time (in seconds) required to complete one trial on the experimental data.

![Comparison of the number of particles between KLD localisation and KLD-Gmapping for the experimental results.]

Figure 5.12: Comparison of the number of particles between KLD localisation and KLD-Gmapping for the experimental results.

<table>
<thead>
<tr>
<th>Application</th>
<th>KLD Localisation</th>
<th>KLD-Gmapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimental Results</td>
<td>47</td>
<td>192</td>
</tr>
</tbody>
</table>

Table 5.4: Computation time (in seconds) for the experimental application of KLD localisation and KLD-Gmapping.

A larger number of particles for KLD localisation was set compared to KLD-Gmapping as the estimation accuracy was unacceptably poor with fewer particles. This is demonstrated by a smaller $\epsilon$ and smaller $\delta$ selected in
Figure 5.11: Comparison of the RMSEs of KLD-Gmapping, KLD localisation and the simulated inertial navigation data.
Table 5.3 for KLD localisation. Even so, the lower RMSE results in Figure 5.11 show that KLD-Gmapping provides a much more accurate estimation than KLD localisation. It is possible that the simulated IMU drift bias $B$ causes more problems for KLD localisation whereas it can be more easily overcome by the maximum likelihood optimisation step of KLD-Gmapping. Both particle filters show a degree of robustness towards the net which was part of the environment/map, but not detected by the LIDAR measurements. However, the localisation results show that the KLD-Gmapping algorithm is potentially more robust towards these errors. Figure 5.12 shows that KLD-Gmapping uses less particles than KLD localisation and achieves a higher estimation accuracy. Although KLD-Gmapping uses approximately ten times fewer particles than KLD localisation, KLD-Gmapping still overall requires much more computation time in comparison.

5.6 Conclusion

This chapter studies the scenario of a robot carrying a LIDAR in a grid-based map. Two important areas of research for localisation are addressed: estimation accuracy and computational demands. These two areas of research are addressed using two influential algorithms: the Gmapping proposal distribution to increase accuracy, and KLD-sampling to reduce computational demands. This chapter’s contribution is that it shows that the two algorithms can be successfully combined to solve the mobile robot localisation problem. The resulting KLD-Gmapping algorithm inherits useful properties from its two constituent algorithms: it is computationally efficient when compared to the standard Gmapping with fixed number of particles and is more accurate than KLD localisation which does not incorporate the recent measurement data into the proposal distribution. The algorithm has been validated in simulation and on a set of experimental data. The experimental data consisted of actual LIDAR data with simulated IMU data based on the ground-truth pose determined by an OptiTrack camera system. The algorithm’s estimation accuracy was compared against the ground-truth pose. The experimental data contained two additional challenges for the algorithms: a bias was added to the simulated IMU data and a net that formed part of the map/environment was not detected in the LIDAR measurement set. Both algorithms showed a degree of robustness towards these additional sources of uncertainty, with KLD-Gmapping showing better pose
estimation than KLD localisation. KLD-Gmapping is a promising solution for problems where a high standard of localisation accuracy is required and the computational requirements are a concern, but this concern is not critical, since KLD-Gmapping still requires quite a large amount of computation resources.
Chapter 6

Conclusion and Future Work

The mobile robot localisation problem consists of estimating the pose of a mobile robot using non-GPS sensors with a given map of the environment. Solving this problem is useful in environments without GPS, for example: indoor, underground, underwater or extra-terrestrial environments. Some work required in these environments could be dangerous for humans. Humans can be replaced by robots to perform some tasks, given that the robot is aware of its own pose in the environment. This thesis examines various aspects of mobile robot localisation including: sensors, proposal distributions and computation.

Chapter 3 and Chapter 4 propose that a Doppler-azimuth radar can be used as an exteroceptive sensor to solve the mobile robot localisation problem in a feature-based map, which is typically used to model sparse environments such as a large, empty warehouse with a few detectable items scattered throughout. Using Doppler-azimuth radars instead of other sensors may result in autonomous robots that are cheaper, lighter and use less power.

Chapter 3 relies on a number of simplifying assumptions about the Doppler-azimuth radar. It is assumed that: all landmarks are hit by the radar, the radar does not produce any false detections, all landmark-measurement associations are known, the Doppler-azimuth radar noise is Gaussian and that the distribution of the robot’s initial pose is known. Following these assumptions, Chapter 3 demonstrates that it is feasible to use a Doppler-azimuth radar as an exteroceptive sensor for mobile robot localisation as demonstrated by a low Cramer-Rao lower bound which is theoretically, the best achievable second-order error performance for nonlinear filtering independent of measurements. Chapter 3 shows in simulation, that an extended Kalman filter is
able to solve the mobile robot localisation problem under these assumptions, demonstrated by its estimation accuracy. The number of landmarks in the environment is also studied which shows that as the number of landmarks increase, the Cramer-Rao lower bound strictly decreases which is intuitive because a greater availability of information should strictly lead to a better theoretical accuracy. The estimation accuracy achieved by an extended Kalman filter generally improves as the number of landmarks increases, but does not strictly improve. This is possibly due to effects of random noise. The findings of this study were published by Guan et al. [6], [9]. Chapter 3 also introduces the use of the SIR particle filter as well as the state-of-the-art Exact Daum-Huang particle flow particle filter. Both filters are shown to be suitable algorithms to solve the mobile robot localisation problem assuming a Doppler-azimuth radar. The simulation results show that the SIR particle filter has the best estimation quality and the lowest number of failures. However, the SIR particle filter consumes the greatest computational resources, by far. In comparison, the extended Kalman filter is by far, the fastest algorithm to run and it demonstrates satisfactory performance.

Building upon the assumptions made in Chapter 3, Chapter 4 models the Doppler radar using more realistic assumptions. Chapter 4 assumed that: landmarks can be missed by the Doppler-azimuth radar (modelled by a probability of detection), false (spurious) detections/clutter can occur (both the number of false detections and the value of the false detection are random), unknown landmark-measurement associations (it is not known from which landmark a measurement has been produced from, if any), and that the robot’s initial pose is unknown. The proposed solution to this problem is the use of a random finite set particle filter. The random finite set particle filter considers all possible hypotheses over the different combinations of hits, misses, false detections and landmark-measurement associations (the only hypothesis excluded from the random finite set particle filter is the hypothesis where all landmarks are assumed to be missed). However, this list of hypotheses will grow exponentially with the number of landmarks. As part of the proposed solution, Murty’s $k$-best assignment algorithm is applied to restrict the number of hypotheses to only the $k$-most likely ones, therefore reducing the computational resources required to run the random finite set particle filter. Furthermore, the number of particles in the particle filter is adaptively determined by the KLD-sampling algorithm which further reduces computational requirements. The simulation results from Chapter 4 show that under more realistic assumptions, the random finite set particle
filter combined with Murty’s algorithm and KLD-sampling is a reasonable solution to the mobile robot localisation problem in a feature-based map, as demonstrated by its estimation accuracy of the robot’s pose. It is also demonstrated that the KLD-sampling algorithm can be applied to the problem involving the realistic noise model for the Doppler-azimuth radar. The findings in this chapter were published by Guan et al. [7].

Instead of a feature-based map, Chapter 5 studies localisation using an occupancy grid-based map which is typically used to represent dense, indoor environments, for example, an office. Chapter 5 also assumes the use of LIDAR as the exteroceptive sensor. The LIDAR is modelled as an array of laser beams with a probability distribution made up of a weighted combination of four other probability densities representing sources of noise. These constituent noise distributions are: Gaussian (when an object is successfully hit), exponential (when an object has obstructed the beam forming an unusually short measurement), point-mass (when a reflected laser beam is not received) and uniform (representing unexplainable, random clutter). This chapter introduces “Gmapping” which is an influential, improved proposal distribution relying on numerical optimisation to propose particles around the maximum likelihood of the received measurement. A contribution of this chapter is combining Gmapping with KLD-sampling to produce the “KLD-Gmapping” algorithm. As discussed previously, KLD-sampling is an algorithm that adapts the number of particles in a particle filter to reduce computational requirements. KLD-Gmapping was applied in simulation and also applied to a set of experimental data. The results show that KLD-Gmapping is very accurate and requires a low number of particles compared to conventional KLD localisation. Despite using an adaptive number of particles, KLD-Gmapping is still very computationally expensive. This chapter demonstrates that KLD-sampling can be successfully combined with different proposal distributions. The findings of this chapter were published by Guan et al. [10],[8].

Future research could examine: localisation using uncertain maps, active localisation - where the robot’s trajectory is optimised to increase estimation accuracy, multi-robot cooperative localisation, localisation using other sensors, or an extension into 3-D space involving a robot such as an unmanned aerial vehicle. Furthermore, localisation experiments could be conducted using actual Doppler-azimuth radar hardware.
CHAPTER 6. CONCLUSION AND FUTURE WORK
Bibliography
Bibliography


[93] Tiancheng Li, Shudong Sun, Tariq Pervez Sattar, and Juan Manuel Corchado. “Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches”. In: Expert Systems with


156

BIBLIOGRAPHY


