Towards Adaptive Sensor Fusion for Simultaneous Localization and Mapping

Akshay Rao
School of Electrical and Electronics Engineering
Nanyang Technological University

A thesis submitted to the Nanyang Technological University in partial fulfilment of the requirement for the degree of

Doctor of Philosophy
I would like to dedicate this thesis to my loving parents.
Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains less than 65,000 words including appendices, bibliography, footnotes, tables and equations and has less than 150 figures.

Akshay Rao
2016
Acknowledgements

At the outset I would like to acknowledge my supervisor, Professor Wang Han, without whose guidance and support, this thesis would not have been completed. I would also like to express my sincere gratitude to Professor Martin Adams, who introduced me to the concept of SLAM and sensor fusion, and whose simple explanation of the sensor fusion algorithms stood me in good stead while completing this thesis. I would also like to sincerely thank the late Professor Wijesoma who discussed the concept of adaptive state estimation with me, and whose untimely death left all his students in deep sorrow.

I would like to thank Dr. John Mullane for his insight and introduction to SLAM algorithms, and his guidance in obtaining and incorporating external data for hypothesis testing and verification. I am grateful for Dr. Diluka Moratuwage for his patience and insight into autonomous robotics, as well as Dr. Namal Senarathne for all those long and interesting discussions we shared on a wide variety of topics.

I would also like to offer my sincere thanks to Professor Mohan Rajesh Ellara for his help and support for my research, and for sharing his vision of new avenues of autonomous robotic research.

Last, but not least, I would like to express my sincere gratitude to my parents for their help, support and patience during my PhD, without which I could definitely not have been able to complete my research.
Abstract

Autonomous vehicles have been used in a variety of environments which are too hazardous or difficult for a human to operate in safely and reliably over an extended period of time. Among their many applications, exploration missions in dynamic and unstructured environments with noisy measurements are quite commonplace.

In the absence of apriori information about the environment, and uncertain localization information, Simultaneous Localization and Mapping (SLAM) algorithms are employed to improve navigation and mapping accuracy. SLAM algorithms accept measurements from proprioceptive sensors, and exteroceptive sensors, independent of each other. These measurements are assumed to be corrupted by noise having Gaussian characteristics with zero bias and a known variance. The measurements obtained are probabilistically fused together using a sensor fusion algorithm.

Robotic platforms are increasingly being deployed in a wide spectrum of environments with dynamics which are too complicated or impossible to model apriori. Common examples include Unmanned Aerial Vehicles (UAVs) being used for surveillance of windy environments, or Autonomous Underwater Vehicles (AUVs) deployed for mapping the seabed.

The prevalence of unmodelled environmental biases, for instance, sudden gusts of wind for UAVs, or the wake of passing boats for AUVs, complicate the deployment of the robotic system from a laboratory environment to a more complicated real world scenario. Sucess-
ful formulation and implementation of algorithms in laboratory settings may not always translate to success during deployment in real world settings.

Furthermore, the increase in prevalence of applications demanding long term robot autonomy ensures that the robotic platform will face degradation of hardware due to wear-and-tear over sustained usage. For example, deployment of Autonomous Surface Crafts (ASCs) in marine environments with saline water will result in the motor being encrusted with brine over long periods of time, causing a change in the motion model characteristics, resulting in divergence of pose and map estimates over long intervals of time.

Present localization and mapping algorithms are unable to face these challenges in their present form. Correction of state estimates due to extremely accurate exteroceptive sensors may prove sufficient for deployment for short durations of time, but the computational cost incurred due to the feature extraction and data association modules to improve accuracy, makes them a poor choice for most applications demanding long term deployment.

Consequently, a more suitable solution for such scenarios is an algorithm incorporating an interoceptive module to increase adaptivity to changing environments by integrating incoming observations from unaffected sensors, in this case, the exteroceptive sensor.

Wrongly estimated noise statistics or mismodeled interoceptive models can cause divergence in the state estimate, regardless of the sensor fusion algorithm used, as this dissertation will show. Consequently, the deployment of adaptive sensor fusion based SLAM algorithms, that adapt to changing model noise statistics, is necessitated.

This thesis first explores the usage of the Kalman Smoother algorithm to accurately estimate the motion model covariance matrix for the Extended Kalman Filter (EKF). While an initially promising candidate, further analysis reveals its unsuitability to form as the basis of an interoceptive algorithm due to inherent flaws in the formulation.

This thesis then reformulates the existing EKF-SLAM and the Factorized Solution to SLAM (FastSLAM) algorithms using the Adaptive Limited Memory Filter (ALMF) into
the Adaptive Extended Kalman Filter-SLAM (AEKF-SLAM) and the Adaptive Extended Kalman Filter-FastSLAM (AEKF-FastSLAM) algorithms. The AEKF-SLAM iteratively estimates the process model noise covariance, while the AEKF-FastSLAM iteratively estimates both the process model noise covariance, as well as the bias.

Performance of the proposed algorithm is evaluated in a marine environment and further performance gains are obtained by considering unmodeled environmental biases caused by ocean currents or adverse field effects such as those due to the wake of passing ships and boats.

The formulations based on the ALMF are both found to have an over-reliance on window size. This drawback is analyzed through extensive simulations which display the variance in performance of the algorithms for different window sizes.

An alternative approach using the Gaussian Particle Filter (GPF) is then examined. The prediction density of the Gaussian Particle Filter is favourably compared with the prediction densities of the Extended Kalman Filter (EKF), the particle filter, the Unscented Kalman Filter (UKF) and the Central Difference Kalman Filter (CDKF) using the Kolmogorov Smirnov statistic.

The Gaussian Particle Filter is then used to formulate the Gaussian Particle Filter SLAM (GPF-SLAM) algorithm, and the estimation errors are compared with the errors obtained from the EKF-SLAM, UKF-SLAM and FastSLAM algorithms. The comparison is done using a simulated trajectory, as well as data obtained from marine environment.

An Adaptive Gaussian Particle Filter based FastSLAM (AGPF-FastSLAM) algorithm is then formulated in which the Gaussian Particle Filter is used to evolve the noise statistics for the FastSLAM prediction density. The bias and covariance of the process model noise is calculated at each step, and fed back into the algorithm during the next step. This approach is found to outperform the EKF-SLAM, FastSLAM and UKF-SLAM algorithm, as well as the AEKF-FastSLAM algorithm discussed previously.
This dissertation is then concluded, and avenues for future research in adaptive sensor fusion for SLAM discussed.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>xiii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xvii</td>
</tr>
<tr>
<td>Nomenclature</td>
<td>xxiv</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Objective</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Achievements</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Organization of the Thesis</td>
<td>6</td>
</tr>
<tr>
<td>2 Literature Review</td>
<td>9</td>
</tr>
<tr>
<td>2.1 Adaptive Kalman Filter</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Kalman Filter Bias Estimation</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Simultaneous Localization And Mapping (SLAM)</td>
<td>16</td>
</tr>
<tr>
<td>2.3.1 A Bayesian Approach to the Joint SLAM Problem</td>
<td>17</td>
</tr>
<tr>
<td>2.3.2 Popular Bayesian SLAM Algorithms</td>
<td>20</td>
</tr>
<tr>
<td>2.3.3 Gaussian Particle Filter based SLAM</td>
<td>24</td>
</tr>
<tr>
<td>3 Moving Window based Approaches to SLAM with Online Compensation of Changes in Motion Model Noise Covariance</td>
<td>27</td>
</tr>
</tbody>
</table>
3.1 Motivation ................................................................. 27
3.2 The Kalman Smoother based EKF SLAM .......................... 30
  3.2.1 Introduction to the Kalman Smoother ......................... 30
  3.2.2 Simulation ......................................................... 32
  3.2.3 Results .......................................................... 35
  3.2.4 Drawbacks of the Kalman Smoother .......................... 36
  3.2.5 Conclusion ....................................................... 38
3.3 Adaptive Limited Memory Filter for Nonlinear Models .......... 39
3.4 The Adaptive Extended Kalman Filter Simultaneous Localization and Mapping (AEKF-SLAM) Algorithm ..................... 40
  3.4.1 Formulation ....................................................... 40
  3.4.2 Simulation Results ............................................... 43
  3.4.3 Experimental Results ........................................... 43
3.5 Conclusion ............................................................. 46

4 Extension of Moving Window Approach to Bias Compensation and Drawbacks Encountered ................................................. 49
4.1 Motivation ............................................................. 49
4.2 ALMF with Bias Estimation - The AEKF-FastSLAM Algorithm .... 51
  4.2.1 Formulation ....................................................... 52
  4.2.2 Experimental Results ............................................ 53
4.3 Drawbacks of the Moving Window Approach ........................ 53
  4.3.1 Simulation ......................................................... 54
  4.3.2 Results .......................................................... 59
4.4 Conclusion ............................................................. 63
5 A Comparison of SLAM Prediction Densities using the Kolmogorov Smirnov Statistic

5.1 Introduction ......................................................... 65
5.2 Formulation ......................................................... 67
  5.2.1 Motion Model Noise Statistics ............................... 67
  5.2.2 Prediction Density Approximation ......................... 68
  5.2.3 Prediction Density Generation Equations .................. 70
  5.2.4 Kolmogorov Smirnov Statistic Computation ............... 75
  5.2.5 Pseudocode to Compute the Kolmogorov-Smirnov Statistic 76
5.3 Simulation ......................................................... 77
5.4 Simulation Result Analysis ...................................... 78
  5.4.1 Effect of Change in Inputs .................................. 78
  5.4.2 Effect of Change in Standard Deviations .................. 81
  5.4.3 Effect of Change in Number of Gaussians for the Gaussian Particle Filter ................................. 83
5.5 Experimental Results .......................................... 83
5.6 Conclusion ....................................................... 88

6 A Gaussian Particle Filter based Factorised Solution to the Simultaneous Localisation And Mapping problem

6.1 Introduction ....................................................... 91
6.2 Formulation ....................................................... 93
6.3 Implementation ................................................... 96
6.4 Experimental Results .......................................... 100
6.5 Conclusion ...................................................... 102
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7.1</td>
<td>Introduction</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>7.2</td>
<td>Formulation</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>7.3</td>
<td>Simulation</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>7.4</td>
<td>Experimental Results</td>
<td>113</td>
</tr>
<tr>
<td></td>
<td>7.5</td>
<td>Conclusion</td>
<td>115</td>
</tr>
<tr>
<td>8</td>
<td>8.1</td>
<td>Conclusions</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>8.2</td>
<td>Future Work</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td></td>
<td>References</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Appendix A</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Derivation for optimal Covariance</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Appendix B</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Publication List</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B.1</td>
<td>Journal Papers</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td>B.2</td>
<td>Conference Papers</td>
<td>131</td>
</tr>
</tbody>
</table>
List of Figures

3.1 Hardware Degradation due to wear-and-tear from long term operation may result in change in covariance of vehicle noise statistics. An example is the degradation of the propeller of a marine vehicle due to extended operation in saline water. .............................. 28

3.2 Comparison of RMS errors for different ratios of real and model standard deviations taken from [23]. The Y-axis measures RMS error, while the X-axis displays the ratio of modelled standard deviation to true standard deviation. The three different curves are characteristics for different number of state estimates, given by $n$. ......................... 28

3.3 Robot Motion Model used. The turn angle is marked as G. The Wheelbase size is WB ......................................................... 32

3.4 Simulated Trajectory. The Trajectory taken by the vehicle shown in blue, the vehicle is represented as a red triangle. The real features are displayed as green asterisks. The estimated landmarks are shown as red points. The feature associations for the current scan are shown as yellow lines .... 33

3.5 RMS Error for case 1. $Q_{actual} = Q_{model}$ ........................................... 34

3.6 RMS Error for case 2. $Q_{actual} = Q_{model}/1.3$ ................................. 35

3.7 RMS Error for case 3. $Q_{actual} = 1.3Q_{model}$ ..................................... 37
3.8 RMS Error comparisons for three different rates of input. The inputs for the plot labelled 'Trajectory 2' is 10 times the inputs for the plot for 'Trajectory 1'. The inputs for the plot labelled 'Straight Line Path' is 0.1 times the inputs for the plot for 'Trajectory 1', making the path the most linear of the three scenarios. .......................................................... 37

3.9 RMS Error for case 1. \( Q_{actual} = Q_{model} \) ........................................ 42

3.10 RMS Error for case 2. \( Q_{actual} = Q_{model}/1.3 \) ................................. 42

3.11 RMS Error for case 3. \( Q_{actual} = 1.3 Q_{model} \) .................................... 43

3.12 Experimental Setup used to gather Data .................................................. 44

3.13 Sample Scan obtained, overlayed on Google Earth image for reference. ... 44

3.14 Trajectory obtained via EKF SLAM (Green: Ground truth, Blue: EKF estimate, Blue crosses: Estimated landmarks, Yellow lines: Superimposed observations, Red lines: Data Association) ................................. 46

3.15 Trajectory obtained via AEKF SLAM (Green: Ground truth, Blue: AEKF estimate, Blue crosses: Estimated landmarks, Yellow lines: Superimposed observations, Red lines: Data Association) ................................. 47

3.16 Comparison of AEKF SLAM and EKF SLAM MS Errors (Blue: MS Error due to EKF-SLAM, Red: MS Error due to AEKF-SLAM) .................. 48

4.1 Environmental causes for bias may be unmodelled gusts of wind (left), or change in expected odometry slip due to navigation along an incline (right) 49

4.2 Trajectory through standard FastSLAM (Green: Ground truth, Blue: FastSLAM, Blue crosses: Estimated landmarks, Yellow lines: Superimposed observations, Red lines: Data Association) .......................... 54
4.3 Trajectory using AEKF-FastSLAM, without bias estimation (Green: Ground truth, Blue: SLAM estimate, Blue crosses: Estimated landmarks, Yellow lines: Superimposed observations, Red lines: Data Association) 55

4.4 Trajectory using AEKF-FastSLAM with bias estimation (Green: Ground truth, Blue: AEKF-FastSLAM estimate, Blue crosses: Estimated landmarks, Yellow lines: Superimposed observations, Red lines: Data Association) 56

4.5 Mean Square Error Comparison (Magenta: FastSLAM RMS error, Red: RMS Error of AEKF-FastSLAM without bias estimation, Blue: RMS Error of AEKF-FastSLAM with bias estimation, Blue: RMS Error of EKF-SLAM) 57

4.6 Simulated trajectory of the platform. The red and green trajectory lines depict the presence of simulated bias for simulations analyzing the effects of bias. The green line represents the inclusion of additive bias (simulating motion downslope or in the direction of wind), while the red line represents subtractive bias (simulating motion upslope or against the direction of wind) 58

4.7 Estimation without noise 59

4.8 Estimation with fixed drift 60

4.9 Estimation with random drift 61

4.10 Estimation with model covariance larger than actual covariance 62

4.11 Estimation with model covariance smaller than actual covariance 63

5.1 Hardware Setup. The robot platform contains sonars, panoramic and stereo cameras, a GPS sensor, one 2D planar LiDAR and one back-and-forth LiDAR 66
5.2 Simulated Trajectory. Axes dimensions are in metres. The robot is represented by a red triangle, and its trajectory is displayed by red *'s. The particle filter hypotheses are displayed as cyan dots. Covariances are displayed as ellipses, while estimates are displayed as +s. EKF predictions are in blue, GPF in black, UKF in green and CDKF in yellow.

5.3 The particle ordering scheme for the calculation of the Kolmogorov-Smirnov statistic is explained in this figure. The trajectory is displayed with cyan asterisks, the current pose is in blue colour and the generated particles are in red colour. 5 particles are simulated for illustration purposes, with their ordering displayed in text alongside the respective particles. The particles are labeled in order of signed angle change. In case of two particles with the same angle change, the particle velocities are used as tie-breaker.

5.4 Partial simulated trajectory. Axes dimensions are in metres. The robot is represented by a red triangle, and its trajectory is displayed by red *'s. The particle filter hypotheses are displayed as cyan dots. Covariances are displayed as ellipses, while estimates are displayed as +s. EKF predictions are in blue, GPF in black, UKF in green and CDKF in yellow. Number of particles displayed is fewer than number used for simulation, to allow greater clarity in figure.

5.5 Velocity and angle inputs are increased simultaneously and the Kolmogorov-Smirnov statistic is calculated for each iteration. The x-axis displays a multiplicative factor for a base velocity of 0.05m/sec and a base angle change of 0.1rad. The vehicle model covariances are $\sigma_v^2 = 0.4m^2/s^2$ and $\sigma_{\delta\theta}^2 = 0.1rad^2/s^2$. 

---

69

73

78

79
5.6 Only Velocity input is increased and the Kolmogorov-Smirnov statistic is calculated for each iteration. The x-axis displays a multiplicative factor for a base velocity of $0.05m/sec$. The vehicle model covariances are $\sigma_v^2 = 0.4m^2/s^2$ and $\sigma_{\delta\theta}^2 = 0.1rad^2/s^2$.

5.7 Only angle input is increased and the Kolmogorov-Smirnov statistic is calculated for each iteration. The x-axis displays a multiplicative factor for a base angle change of $0.1rad$. The vehicle model covariances are $\sigma_v^2 = 0.4m^2/s^2$ and $\sigma_{\delta\theta}^2 = 0.1rad^2/s^2$.

5.8 Change in Kolmogorov-Smirnov Statistic when velocity and angle standard deviation are simultaneously changed. The x-axis displays a multiplicative factor for a base velocity standard deviation of $0.5m/s$ and a base angle change standard deviation of $0.4rad$. $N_{GPF} = 50$.

5.9 Change in Kolmogorov-Smirnov Statistic when only velocity standard deviation is changed. The x-axis displays a multiplicative factor for a base velocity standard deviation of $0.5m/s$. $N_{GPF} = 50$.

5.10 Change in Kolmogorov-Smirnov Statistic when only angle change standard deviation is changed. The x-axis displays a multiplicative factor for a base angle change standard deviation of $0.4rad$. $N_{GPF} = 50$.

5.11 Change in Kolmogorov-Smirnov Statistic when number of Gaussians for the Gaussian particle filter prediction density is changed. The x-axis displays a multiplicative factor for a base $N_{GPF} = 3$. The vehicle model covariances are $\sigma_v^2 = 0.4m^2/s^2$ and $\sigma_{\delta\theta}^2 = 0.1rad^2$. 
5.12 Trajectory of robot in experiment. The figure shows a map obtained from the ROS Hector mapping package. The blue arrows depict the trajectory predicted by the visual odometry, while the red arrow depicts the current ground truth, obtained from the Hector mapping estimate. The background grid has a unit size of $1m$.

5.13 Prediction density likelihood obtained from experiment. The x-axes displays a multiplicative factor with a base velocity of $0.1m/s$ and angle change of $\pi/50$.

5.14 Trajectory of robot in experiment with no angle change. The figure shows a map obtained from the ROS Hector mapping package. The blue arrows depict the trajectory predicted by the visual odometry, while the red arrow depicts the current ground truth, obtained from the Hector mapping estimate. The background grid has a unit size of $1m$.

5.15 Prediction density likelihood obtained from experiment with no angle change. The x-axes displays a multiplicative factor with a base velocity of $0.1m/s$ with no angle change.

5.16 Trajectory of robot in experiment with high angle change. The figure shows a map obtained from the ROS Hector mapping package. The blue arrows depict the trajectory predicted by the visual odometry, while the red arrow depicts the current ground truth, obtained from the Hector mapping estimate. The background grid has a unit size of $1m$.

5.17 Prediction density likelihood obtained from experiment with small angle change. The x-axes displays a multiplicative factor with a base velocity of $0.04m/s$ and angle change of $\pi/50$. 
6.1  Simulated Trajectory. The Trajectory taken by the vehicle shown in black, 
the vehicle is represented as a green triangle. The real features are displayed 
as blue crosses. The current scan associations are shown as red lines. . . . . 97

6.2  Comparison of Trajectory Errors after 50 Simulated Runs . . . . . . . . 98

6.3  Comparison of Computational Time after 50 Simulated Runs . . . . . . 99

6.4  Localization Error Comparison . . . . . . . . . . . . . . . . . . . . . . . 101

6.5  Computational Time Comparison using Data from Kayak . . . . . . . . 101

7.1  Simulated Trajectory, with the axes marked in metres. The green asterisks 
are the feature location ground truths, the red dots adjacent to the aster-
isks are the feature location estimates. The red triangle is the vehicle pose 
ground truth, while the adjacent green triangle is the pose estimate. Addi-
tive bias is added to the noise in the part of the trajectory shown in green 
colour, while subtractive bias is added to the noise in the part of the trajec-
tory shown in red colour . . . . . . . . . . . . . . . . . . . . . . . . . . . 107

7.2  Initial distribution of estimated map with newly initialized process model 
gaussian hypotheses . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 108

7.3  Converged distributions of estimated map with converging process model 
gaussian hypotheses . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 108

7.4  Estimation without noise . . . . . . . . . . . . . . . . . . . . . . . . . . . 110

7.5  Estimation with fixed drift . . . . . . . . . . . . . . . . . . . . . . . . . . 111

7.6  Estimation with random drift . . . . . . . . . . . . . . . . . . . . . . . . 111

7.7  Estimation with model covariance larger than actual covariance . . . . . 112

7.8  Estimation with model covariance smaller than actual covariance . . . . . 112

7.9  RMS Error results of Experimental Data from Kayak . . . . . . . . . . 114
8.1 Estimation of angle standard deviation when initial model standard deviation is greater than actual standard deviation. . . . . . . . . . . . . . . . . . 120
8.2 Estimation of velocity standard deviation when initial model standard deviation is greater than actual standard deviation. . . . . . . . . . . . . . . . . . 120
8.3 Estimation of angle standard deviation when initial model standard deviation is lesser than actual standard deviation. . . . . . . . . . . . . . . . . . 121
8.4 Estimation of velocity standard deviation when initial model standard deviation is lesser than actual standard deviation. . . . . . . . . . . . . . . . . . 121
Chapter 1

Introduction

Simultaneous Localization and Mapping, or SLAM, has been a topic of much research in the robotics community in recent years. The SLAM family of algorithms simultaneously estimates the trajectory of the robot and the map of landmarks around it, conditioned on the observations and control inputs received by the robot thus far. It provides a solution to the equation written below.

\[
p(x_t, m|z_{1:t}, u_{1:t}) = \int \int \ldots \int (t-1) \text{ times} \int p(x_{1:t}, m|z_{1:t}, u_{1:t}) dx_1 dx_2 \ldots dx_{t-1} \quad (1.1)
\]

Where

- \(x_t\) and \(m\) represent the robot pose and map estimates at time \(t\) respectively,
- \(x_{1:t}\) is the set of pose estimates from the first time iteration to the present iteration,
- \(z_{1:t}\) is the set of observations received by the robot from the first time step to the present time step and
- \(u_{1:t}\) is the set of control inputs received by the robot from the first time step to the current
This expression, unfortunately, does not have a closed form solution. However, various approximate solutions using the Extended Kalman Filter, the Unscented Kalman Filter and the Particle Filter among other sensor fusion algorithms exist, and perform satisfactorily in most well-modeled environments.

The motion model and the observation model in SLAM algorithms are assumed to be corrupted with a zero-mean white gaussian noise. Hence, SLAM algorithms, in general, comprise a Kalman Filter core to probabilistically fuse observations of the vehicle pose and the map using a vehicle process model, and an observation model, respectively. In most real world application scenarios, the nonlinearity of the process and observations models lead to the implementation of linearisation schemes such as Extended Kalman Filters, Unscented Kalman Filters, Particle Filters and Ensemble Kalman Filters, depending on the usage scenario. The mitigation of uncertainty and errors due to the employment of linearisation schemes has been the subject of a vast body of work.

The Extended Kalman Filter is the most often used sensor fusion scheme in scenarios involving one or more nonlinear process models. It linearizes the model at the point of calculation for the Kalman Filter using the following equations:

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

\[
P_{k|k-1} = F_k P_{k|k-1} F_k^T + Q_k
\]

\[
K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}
\]
\[ x_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H_k \hat{x}_{k|k-1}) \] (1.5)

\[ P_{k|k} = (I - K_k H_k) P_{k|k-1} \] (1.6)

where

\( \hat{x}_{k|k-1} \) and \( P_{k|k-1} \) are the predicted state and covariance, respectively, 

\( K_k \) is the Kalman Gain 

\( \hat{x}_{k|k} \) and \( P_{k|k} \) are the predicted state and covariance, respectively, 

\( z_k \) is the observation vector at time \( k \) 

and \( Q_k \) and \( R_k \) are the vehicle process noise covariance, and the observation noise covariance respectively.

Sources of uncertainty in SLAM arising from the observation model for the exteroceptive sensor are data association modules, feature extraction modules and map management modules, all of which have been the subject of a variety of formulations aiming to minimise the consequent uncertainty.

All solutions for SLAM suffer from a major source of error arising from the use of the Kalman Filter, namely, the misestimation of the process and observation model variances (\( Q_k \) and \( R_k \) respectively), due to the assumption of a constant interoceptive vehicle model. Divergence in either statistic due to poorly formulated apriori models, or unmodeled changes due to external environmental factors, will cause a ripple-effect resulting in divergence in long-term state estimates as this dissertation will demonstrate. In addition, unlike most SLAM implementation scenarios, the increase in errors will remain unmitigated even after loop closure.

Much research has been performed on the online estimation of these statistics and ap-
PLIED TO VARIOUS FIELDS, AS DIVERSE AS POWER ELECTRONICS AND COMMUNICATIONS, BUT HAS YET TO BE INTRODUCED TO THE FIELD OF SLAM. THE APPLICATION MOST SIMILAR TO SLAM WITH A FORMULATION CONTAINING THE ONLINE ESTIMATION OF THE NOISE COVARIANCE STATISTICS, IS THAT OF VEHICLE LOCALISATION USING GPS OBSERVATIONS.

THE POSSIBILITY OF EXTENDING THESE ALGORITHMS TO REFORMULATE KEY, OFTEN-USED SLAM ALGORITHMS HAS BEEN INVESTIGATED SUCCESSFULLY SO FAR. INHERENT DRAWBACKS DUE TO IMPLEMENTATION OF THESE ALGORITHMS WILL BE EXAMINED AND LEAD TO A MORE COMPREHENSIVE FORMULATION FOR ESTIMATION OF THE INTEROCEPTIVE MODEL PARAMENTERS.

1.1 Objective

The main objective of the research work is to develop algorithms to solve the Simultaneous Localization And Mapping (SLAM) problem using an adaptive sensor fusion formulation of standard SLAM algorithms, so as to improve navigation of an autonomous vehicle in dynamic, uncertain and unstructured environments, with rapidly changing, unmodelled noise characteristics.

More specifically it involves developing generic algorithms for an autonomous robot using some of a wide variety of sensors and actuators capable of gathering information under field conditions, without any constraints on the noise characteristics affecting each sensor, while overcoming limitations of conventional algorithms such as apriori modeled sensor and vehicle process models which are incapable of adapting to externally induced changes due to unmodelled changes in environment, or degradation due to wear-and-tear of hardware.
1.2 Achievements

SLAM was first performed in a highly unstructured marine environment, with the estimates performed despite nonlinear motion and clutter filled observations, and the results were favourably compared to a scheme comprising vehicle localization using a GPS receiver, and mapping using superimposed observations.

A well-known SLAM algorithm using static vehicle process models was then extended to incorporate an Adaptive Kalman Filter. While providing a noticeable performance gain, this approach suffered from the curses of dimensionality and nonlinearity.

To address these issues, an improved Adaptive Kalman Filter based SLAM algorithm was developed by combining a particle filter and an extended Kalman Filter using the Fast-SLAM algorithm. Although expected performance gains were achieved, it suffered from un-modelled biases corrupting the estimation. Hence, a bias estimation component was added to the algorithm, causing major improvements in estimates.

The SLAM algorithms formulated were found to exhibit an over-reliance on the window size, thus an alternative solution utilizing the Gaussian Particle Filter was explored. Prediction density approximations of a multitude of state estimation schema were compared using the Kolmogorov-Smirnov statistic and the Gaussian Particle Filter was found superior to all the schema compared.

Consequently, a Gaussian Particle Filter based SLAM algorithm was successfully implemented and the results were favourably compared to other state of the art SLAM algorithms. The Gaussian Particle Filter was then employed in the formulation of an adaptive SLAM algorithm with regular updates of the motion model noise statistics.
1.3 Organization of the Thesis

The remainder of the thesis is organized as follows. Chapter 2 provides an overview of state of the art solutions for SLAM, Adaptive Kalman Filtering and the Gaussian Particle Filter.

Chapter 3 explores the formulation of a SLAM algorithm with a module for compensation of changes in covariance. An initial attempt is made by formulating an adaptive Extended Kalman Filter (EKF) SLAM algorithm using the Kalman Smoother. The resultant algorithm is evaluated using a simulation under a wide spectrum of conditions for motion model noise statistic mismodelling, and favourably compared to the EKF-SLAM algorithm. Drawbacks revealed after further analysis of the algorithm are then analyzed. An alternative approach using the Adaptive Limited Memory Filter (ALMF) is described. A formulation of the ALMF for the EKF-SLAM algorithm, the Adaptive Extended Kalman Filter SLAM (AEKF-SLAM), is derived and compared favorably against the previously described Kalman Smoother based approach.

Chapter 4 extends the ALMF based algorithm to include compensation of environmental bias. The AEKF-FastSLAM algorithm is formulated using the ALMF algorithm in conjunction with the Factorised Solution to SLAM (FastSLAM). The resultant algorithm is benchmarked using experimental data. Drawbacks of the moving window approach are then explored and analyzed in details through simulations. A possible solution to this drawback is described.

Chapter 5 introduces the Gaussian Particle Filter and favourably compares the prediction density of the Gaussian Particle Filter with other state of the art SLAM algorithms using the Kolmogorov Smirnov statistic. Chapter 6 formulates and describes the successful implementation of a SLAM algorithm based on the Gaussian Particle Filter.

Chapter 7 describes Gaussian Particle Filter based adaptive SLAM algorithm and presents the results obtained via simulation and a marine dataset. Chapter 8 concludes the thesis and
details possible avenues of future research.
Chapter 2

Literature Review

2.1 Adaptive Kalman Filter

The Kalman Filter [33] is an optimal estimator for linear stochastic processes corrupted by white, zero-mean gaussian noise of known covariance. The standard Kalman Filtering equations are as follows:

\[ \hat{x}(k|k-1) = F_k x_{k-1|k-1} + B_k U_k \]  
\[ P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \]  
\[ K_k = P_{k|k-1} H_k (H_k P_{k|k-1} H_k^T + R_k)^{-1} \]  
\[ x_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - H_k \hat{x}_{k|k-1}) \]  

(2.1)  
(2.2)  
(2.3)  
(2.4)
\[
P_{k|k} = (I - K_k H_k) P_{k|k-1}
\]

where
\[
\hat{x}_{k|k-1} \text{ and } P_{k|k-1} \text{ are the predicted state and covariance, respectively,}
\]

\[
K_k \text{ is the Kalman Gain}
\]

\[
\hat{x}_{k|k} \text{ and } P_{k|k} \text{ are the predicted state and covariance, respectively,}
\]

and \( Q_k \) and \( R_k \) are the vehicle process noise covariance, and the observation noise covariance respectively.

In many cases in which it is impractical to obtain noise statistics for the models, appropriately large covariances are assumed. This causes much suboptimality in the estimation which is reflected in the increased Mean Square Error of the estimate, and eventual divergence of the estimate from the ground truth [23].

Noise statistic estimation was first analyzed and commented on in [1]. This paper first observed the consequences of misestimation of noise statistics on Kalman Filter estimation. An optimal estimator was derived based on Maximum Likelihood (ML) estimation, as opposed to Minimum Variance Unbiased Estimation (MVUE). Under the assumption of time-invariant, misestimated noise statistics, a likelihood function was derived for the noise covariance matrices.

\[
\frac{\partial L_n}{\partial \xi} = -1/2 \sum_{i=1}^{n} Tr[(P_i^{-1} - P_i^{-1} \Delta x_i \Delta x_i^T P_i^{-1}) \frac{\partial P_i}{\partial \xi} - 2P_i^{-1} \Delta x_i \frac{\partial \hat{x}_{i|i-1}}{\partial \xi} H_i^T]
\]

where \( L_n \) is the likelihood function,
ξ is a vector of diagonal (non-zero) elements of the noise covariance matrix to be estimated

\( x_i \) is a vector of \( n \) Independent, Identically Distributed (IID) observations of the stochastic variable \( x \)

\( \Delta x_i \) is the residual of the variable at time step \( i \)

\( \xi \) was found as the solution of

\[
\frac{\partial L_n}{\partial \xi} | _{\xi_n} = 0 \quad (2.7)
\]

The statistic was then reconstituted from the elements of \( \xi \).

This method was unfortunately not put to use due to the lack of a closed form solution to the likelihood equation.

The Adaptive Limited Memory Filter (ALMF) was then derived in [53], based on empirical estimators of the noise statistics using state residuals. This algorithm was notable for inclusion of a bias estimation component, along with a noise covariance matrix estimation algorithm. It used a fading window of a user defined number of steps to recursively calculate the covariance matrix for the residuals in said window. It did not assume the covariance matrix to be time-invariant, and hence could be used in situations of stochastic processes affected by time-varying noise.

The equations used to calculate the process noise covariance matrix are as follows:

\[
\Delta x_k = (\hat{x}_k - F_k \hat{x}_{k|k-1}) \quad (2.8)
\]

\[
\Delta Q_k = F_k P_{k|k-1} F_k^T - P_k \quad (2.9)
\]

\[
\hat{q}_k = q_{k-1} + \frac{1}{N} (q_k - q_{k-N}) \quad (2.10)
\]
\[ \hat{Q}_k = Q_{k-1} + \frac{1}{N-1} \sum_{i=1}^{N} [(\Delta x_i - \hat{q}_k)(\Delta x_i - \hat{q}_k)^T - (\Delta x_i - \Delta x_{i-N})(\Delta x_i - \Delta x_{i-N})^T + \frac{1}{N}(\Delta x_i - \Delta x_{i-N})(\Delta x_i - \Delta x_{i-N})^T - \frac{N-1}{N}[\Delta Q_{k-N} - \Delta Q_k]] \]  

(2.11)

where \( \Delta x_k \) is the residual of the variable at time step k, 
\( \Delta Q_k \) is the residual covariance at time k, 
\( \hat{q}_k \) is the window bias at time step k and 
N is the window size.

In [40], the expression calculating the noise covariance matrix \( Q_k \) was further refined to produce the following expression.

\[ \hat{Q}_k = \frac{1}{N-1} \sum_{i=1}^{N} [(\Delta x_i - \hat{q}_k)(\Delta x_i - \hat{q}_k)^T - \frac{N-1}{N} F_k P_{i|i-1} F_k + P_{i|i-1} - (I - K_i H_i) P_{i|i-1} - P_{i|i-1} (I - K_i H_i)^T] \]  

(2.12)

This equation was then simplified in [71] to produce the following result.

\[ \hat{Q}_k = \frac{1}{N-1} \sum_{i=1}^{N} [(\Delta x_i - \hat{q}_k)(\Delta x_i - \hat{q}_k)^T - \frac{N-1}{N} (F_i P_{i|i} F_i - P_{i|i})] \]  

(2.13)

\section{2.2 Kalman Filter Bias Estimation}

Being restricted to linear models suffering from zero-mean noise, efforts were made to reformulate the Kalman Filter to accommodate non-zero noise, as encountered in many real
This aim was first achieved successfully in the algorithm formulated in the seminal paper [22]. The bias in the Kalman filter is accounted for by simply extending the Kalman filter state equations to include the bias terms. With an assumption of accurately known, time-invariant initial bias (such as in the case of misaligned reference frames), this paper proved that subsequent uncertainty in the bias estimation can be reduced to zero. The Kalman Filter equations are modified to include the bias terms as follows:

\[
 x_{k+1|k} = A_k x_{k|k} + B_k b_k + \xi_k \quad (2.14)
\]

\[
 y_{k+1|k} = L_k x_{k|k} + C_k b_k + \eta_k \quad (2.15)
\]

Now, the state estimate and bias estimate are adjoined to create a new state vector.

\[
 z_k = \begin{bmatrix} x_k \\ b_k \end{bmatrix} \quad (2.16)
\]

This results in new process and observation models given by the following equations:

\[
 z_{k+1} = F_k z_{k|k} + G \xi \quad (2.17)
\]

and

\[
 y_{k+1} = H_k z_{k+1|k} + \eta \quad (2.18)
\]
Where

\[
F_k = \begin{bmatrix} A_k & B_k \\ 0 & 0 \end{bmatrix}
\]  
(2.19)

\[
G = \begin{bmatrix} I \\ 0 \end{bmatrix}
\]  
(2.20)

and

\[
H_k = [L_k \ C_k]
\]  
(2.21)

The state covariance matrix is represented by:

\[
P_{k+1|k} = \begin{bmatrix} P_x & P_{xb} \\ P_{xb}^T & P_b \end{bmatrix}
\]  
(2.22)

The Kalman update equations then become:

\[
z_{k+1|k+1} = F_k z_{k+1|k} + P_{k+1|k} H_k^T R^{-1} (y_{k+1} - H_k z_{k+1|k})
\]  
(2.23)

and

\[
P_{k+1|k+1} = F_k P_{k+1|k} F_k^T + P_{k+1|k} H_k^T R^{-1} (y_{k+1} - H_k z_{k+1|k})
\]  
(2.24)
With no uncertainty in the knowledge of the initial bias, the paper proved that $P_{b,k} = 0$ and $P_{sb,k} = 0$ for all $k$. Hence, the above equations devolve to the standard Kalman filtering equations.

The equations obtained from this paper were simplified in [30], and the concept of "sensitivity matrices" were introduced to correlate the bias free and perfectly known bias estimators of the state vector.

If $x_k$ represents the bias-free estimator, and $z_k$ represents the estimator accounting for bias, then

$$z_{k+1|k} = x_{k+1|k} + U_kb_k$$ (2.25)

and

$$z_{k+1|k+1} = x_{k+1|k+1} + V_kb_k$$ (2.26)

where $U_k$ and $V_k$ represent the prior and posterior sensitivity matrices. These were calculated recursively using the following equations:

$$U_k = F_kV_{k-1} + B_k$$ (2.27)

and

$$V_k = U_k - K_k(H_kU_k + C_k)$$ (2.28)
The residual $r_k$ obtained was then used to calculate the bias using the following relation:

$$r_k = (y_k - H_k z_{k+1|k} - C_k b_k) + (H_k U_k + C_k) b_k$$  \hspace{1cm} (2.29)$$

The term $(H_k U_k + C_k) b_k$ represents the measurement residual of the estimator having perfect knowledge of $b_k$. The autocovariance of this term simplifies to the following:

$$E(v_k v_k^T) = H_k P_{x,k+1|k} H_k^T + R_k$$  \hspace{1cm} (2.30)$$

where $v$ has been used to represent the term $(H_k U_k + C_k) b_k$. The RHS of the above equation contains known terms, and hence can be used to calculate the bias through the term $v$. This result also helped prove the equivalence of the filter proposed in [22] and the standard Kalman Filter, in cases of zero bias.

This approach was extended in [72] to include cases of unknown, time-varying bias. This paper made the novel use of the algorithm to encapsulate nonlinear process and observation models, with the nonlinearities modelled as time varying biases.

## 2.3 Simultaneous Localization And Mapping (SLAM)

An autonomous navigation platform is assumed to consist of a mobile platform with a suite of onboard proprioceptive and exteroceptive sensor units. Proprioceptive sensor measurements are used in process models to generate an initial prediction of the vehicle and environmental state. Exteroceptive sensor measurements are then employed to correct the predicted state, while simultaneously extending the map of the environment. Sensor noise and inaccurate readings are the cause of multiple sources of uncertainty in such an autonomous plat-
form. Initial approaches [8], [39] circumvented this problem by using exact measurements, expensive, high-precision sensors and well-spaced points for recalibrating the sensors while in operation. These methods, while sufficiently accurate in their specified limitations, are extremely expensive and fragile in their robustness, while limiting the scope of deployment of the robot.

An alternative approach that has gained popularity in recent years was to algorithmically compensate for the multiple sources of uncertainty by explicitly including them in the formulation of the algorithm. Initial solutions to the SLAM problem included Genetic Algorithm [15] and Kinematic Link [60] based approaches. However, following the seminal work by Smith, Self and Cheeseman [65] which formulated a probabilistic solution to SLAM using a joint Extended Kalman Filtering framework, stochastic approaches gained more popularity. A review of the most popular solutions is presented to better highlight the problems addressed in this thesis.

2.3.1 A Bayesian Approach to the Joint SLAM Problem

A large majority of approaches to SLAM are formulated from approximations to the classical Bayesian SLAM recursion, hence a brief description is provided here.

SLAM formulations assume the vehicle to be moving through an environment containing an unknown distribution of landmarks. At each time iteration $k$, noisy control input $u_k$ is provided to the vehicle to navigate to the next waypoint. A history of all control inputs till the current time step is given by $u^k = [u_0, u_1, ..., u_k]$. The $n_X$ dimensional vehicle state at the current time step is represented as $X_k \in \mathbb{R}^{n_X}$. The vehicle is assumed to begin in a known location $X_0$. The vehicle trajectory till the current time step is denoted as $X^k = [X_0, ..., X_k]$.

The surrounding environment $(M)$ is assumed static and is represented as an $n_M$ dimensional vector. Observations of the map by the exteroceptive sensor at the current time step
is denoted by vector $Z_k \in \mathbb{R}^n$, with $Z^k = [Z_0, ..., Z_k]$ being the history of all measurements till the current time step.

The SLAM problem requires the robot to incrementally construct a map $M$, and localize itself online within the map by estimating its pose $X$, without any a priori information. The stochastic approach to the SLAM problem requires the estimation of a joint probability density function of the map and vehicle location, given by the expression

$$p_{k|k}(X_k, M_k|Z^k, u^k, X_0)$$ (2.31)

Formulating the above expression in a predictor-corrector estimator model, the predicted joint density is given by the equation:

$$p_{k|k-1}(X_k, M_k|Z^{k-1}, u^k, X_0) = \int p_{k|k-1}(X_k|X_{k-1}, u_k)$$

$$p_{k-1|k-1}(X_{k-1}, M_{k-1}|Z^{k-1}, u^{k-1}, X_0) dX_{k-1}$$ (2.32)

The corrected a posteriori estimate is then calculated using the equation:

$$p_{k|k}(X_k, M_k|Z^k, u^k, X_0) = \frac{g(Z_k|M_k, X_k)p_{k|k-1}(X_k, M_k|Z^{k-1}, u^k, X_0)}{\int g(Z_k|M_k, X_k)p_{k|k-1}(X_k, M_k|Z^{k-1}, u^k, X_0) dX_k dM_k}$$ (2.33)

where $g(Z_k|M_k, X_k)$ is the measurement likelihood function.

Most SLAM algorithms are formulated for a single autonomous platform using observations from a single exteroceptive sensor. SLAM algorithms maneuvering in a single plane (such as ground vehicles [16], [54], [18], [11], [13], [70], [67] and [52], or marine platforms [51]), employ a 3 Degree of Freedom (3 DOF) vehicle state containing the Cartesian coordinates of the vehicle, $(x, y)$, along with the vehicle orientation $\theta$. 
2.3 Simultaneous Localization And Mapping (SLAM)

SLAM formulations with state vector having 6 DOF (3 Cartesian coordinates \((x,y,z)\) and 3 Euler angles \((roll,pitch,yaw)\)) are used for aerial [35], underwater [20], and sometimes ground vehicles [6]. This thesis assumes the planar case with \(n_X = 3\). However, the algorithms can be extended to higher dimensional states for non-planar navigational environments.

SLAM formulations assume the presence of static [16] environmental landmarks, which are incrementally mapped without any a priori information regarding the landmark distribution.

Two major mapping approaches are adopted in order to compensate for sensor noise and localization error.

- **Grid based maps:** This approaches discretizes the map state-space into discrete cells defined as per a global Cartesian grid [18],[19], [21], [27], [28], [37], [49], [57], [66], [25]. Each cell is given a value between 0 and 1, depending on the probability of occupancy, where 0 implies complete certainty of the cell being empty, while 1 implies complete certainty of the cell being occupied. The computational complexity involved in maintaining and updating a grid based map makes it a poor choice for modeling environments which are large or non-planar.

- **Feature based maps:** SLAM formulations employ a wide spectrum of possible environmental landmarks depending on the application scenario. For example, bushes or trees for outdoor ground vehicles [26], wall corners for indoor ground vehicles navigating in a plane [43], or buoys or stationary ships in marine vehicles [51], may all be viable landmarks for SLAM.

Each landmark is simplified into a single point-based representation. The map is assumed to contain an unknown number of environmental landmarks of uncertain distribution, both of which must be estimated by the vehicle.
Some popular feature based SLAM formulations include [13], [70], [63], [11], [35], [38], [41], [46], [43], [12], [45] and [44].

2.3.2 Popular Bayesian SLAM Algorithms

Most popular SLAM approaches in recent years have used the Bayesian SLAM formulation described in the previous subsection. Direct solutions of eqs 2.31, 2.32 and 2.33 are computationally intractable, resulting the use of approximations. This section will review a few of the most popular SLAM formulations.

[65] proved that as a mobile robot moves through an unknown environment taking relative observations of landmarks, the estimates of these landmarks are correlated because of the common error in estimated vehicle location. This formulation was then implemented on a land robot in [50]. The algorithm formulated in [65] was Bayes optimal under the strict conditions of unity landmark detection probability, known landmark map measurements, no clutter and no sensor bias.

This approach was further explored in [13] where the idea of using an Extended Kalman Filter (EKF) to probabilistically fuse noisy localization and mapping information was first formulated, and successfully demonstrated. [13] extended the work done in [65] by using observations of the features to increase correlation between all the landmarks, and decrease uncertainty in all the landmark positions, asymptotically ending up with an absolutely precise map, within a tolerance to account for the inherent uncertainty of the sensor.

[13] was extended with the inclusion of data association and feature management sub-algorithms for laser rangefinder [26], Millimeter Wave Radar [11], [10] and sonar [41]. [64] successfully implemented the algorithm on an underwater robot. [17] and [5] reformulated the joint posterior as a Gaussian Mixture (GM) approximation and implemented the algorithm with an underwater sonar.
2.3 Simultaneous Localization And Mapping (SLAM)

Extended Kalman Filter based SLAM and localization formulations assume a proprioceptive sensor affected by uncertainty exhibiting zero-mean Gaussian statistics. Inertial Measurement Units (IMU) used as proprioceptive sensors in Bayesian SLAM algorithms and Inertial Navigation Systems (INS) are affected with inertial bias, resulting in the loss of validity of this assumption. Subsequent algorithms [59], [69], [58], [14], [29] extended the vehicle state in [13] to include the bias variables to explicitly estimate the bias concurrently with the vehicle pose.

While theoretically proven to be sufficient to solve the SLAM problem, [13] relied on several assumptions which are not frequently encountered in practice. For example, non-uniqueness of the features made it necessary to adopt data association algorithms, and commonly-occurring wrong associations caused the state estimation to diverge significantly from the ground truth. Use of the Extended Kalman Filter caused further complications due to linearisation errors encountered when the process model was too nonlinear for the filter to cope with adequately. Computation of the Jacobians for complex nonlinear models also proved to be nontrivial and time-consuming.

In order to better deal with these issues, an initial algorithm was formulated with the use of higher-order Hessian functions [36]. Another, more popular approach employed the formulation and implementation of the seminal Unscented Kalman Filter (UKF) in [32]. The UKF predicted the transformed mean and covariance using the system model equations. For an n-dimensional state vector gaussian, $O(n)$ points or 'scents' are used to encapsulate the information of the state mean and covariance upto the fourth term of the Taylor series. In contrast, the EKF can predict the mean and covariance accurately to the second order of the Taylor series. Apart from its accuracy, the UKF is less computationally intensive compared to the EKF for maps with a large number of landmarks, and has been used in many other fields employing nonlinear process models as well.

The UKF proposed in [32] was able to succesfully overcome the problem the EKF faced
which linearising highly nonlinear models, at much lesser computational cost. However, it was unable to account for many other issues faced while implementing the EKF-SLAM algorithm. While bad data association still remained an issue, the correlation between the elements of the state vector, while previously appreciated for providing better accuracy, became an implementation bottleneck, since the observation of even a single feature resulted in the entire state being updated, requiring time quadratic in the number of features in the map.

In response to these issues, the seminal Factorised Solution to the Simultaneous Localization and Mapping (FastSLAM) algorithm was formulated and implemented in [46] using algorithms developed in the target tracking community [24], [56], [9]. It factorised the SLAM posterior into 2 parts. Localisation was carried out using a particle filter in which each particle was propagated by using the state equations, and sampling from the process model covariance. The particles acted as independent representations of the robot and maintained EKFs representing individual features. The map of each particle was evaluated for the best fit for the current set of observations, and corresponding weights assigned to the particles. The update step consisted of the particles getting resampled according to their weights, and the maps of surviving particles getting updated according to EKF update equations. This approach was also successfully implemented with realtime data association in [55], and in computer vision based SLAM applications in the seminal [34].

The adoption of the particle filter led to the benefit of good convergence of the filter in cases of high nonlinearity as well, while the employment of Extended Kalman Filters to represent individual features solved the issues of bad data association, and over-correlation of features. This was extremely useful in cases of noisy clutter-filled environments, where false detections of features were less likely to disrupt the convergence of the algorithm.

There were, however, many disadvantages arising from the employment of the particle filter, the most important being that the particle filter approximates the pose uncertainty of
the robot asymptotically with the number of particles. Furthermore, even with the assumption of a more accurate process model with less noise, the number of particles required for convergence is exponential to the size of the map.

To overcome this drawback, FastSLAM 2.0 was proposed in [44]. This algorithm made better use of the particles by conditioning the sampling on the previous pose estimate, and the actual current measurement, instead of relying on just the previous pose estimate as was the case with [46]. The weight calculation was also altered to include the new sampling methodology.

One of the most exciting breakthroughs in the field of stochastic state estimation in recent years is the evolution of the Random Finite Set (RFS) approach through the formulation of Finite Set Statistics (FiSSt) developed in [42]. While initially applied to the field of target tracking [68], it was later reformulated to solve the SLAM problem in [31]. The RFS approach assumes the map to be a set of features, as opposed to a vector of features. This approach has the advantage over traditional vector-based algorithms of encapsulating SLAM sub-algorithms, such as data association, map management and clutter filtering into a comprehensive, Bayes optimal formulation.

Like [46], localisation in [31] was performed using a particle filter. Each particle maintained an independent map represented by a Random Finite Set. The first moment of the Random Finite Set, called a Probability Hypothesis Density (PHD) was propagated and updated using an Extended Kalman Filter. The algorithm allowed for the input of user-defined clutter rate and detection probability, which was then used to predict the map. The weights of each particle are calculated as the likelihood of the PHD, conditioned on the previous map, and the predicted pose of the particle. Like [46], after resampling, the PHDs of the surviving particles are updated using the Extended Kalman Filter update equation. [47] and [48] extend this formulation to multi-vehicle solutions.
2.3.3 Gaussian Particle Filter based SLAM

One of the biggest drawbacks of the particle filter, however, is the problem of degeneracy of the particles which occurs when the particles diverge over a long period of time, leading to poor estimates. The resampling sub-algorithm is the solution to this problem. However, sustained resampling causes massive sample impoverishment resulting in a few particles obtaining most of the weight. Due to this, lower weighted competing hypotheses get filtered out, allowing random clutter in observations to cause divergence in estimates [65].

Owing to its dependence on the particle filter, the FastSLAM inherits both these drawbacks [40]. Furthermore, lack of correlation between landmarks decreases convergence rate of the map estimate, unlike the Extended Kalman Filter based SLAM algorithm.

A recently formulated solution to these problems is the Gaussian Particle Filtering algorithm which has been successfully implemented in the target tracking community [32], which is based on the fact that, given the mean and the covariance of any probability distribution, the normal distribution maximizes entropy of the random variable, and is the least informative distribution. The implication of this property is that mean and covariance of any probability density can be tracked using a Gaussian probability distribution function. It has been proven that any probability density can be accurately represented by an infinite weighted Sum of Gaussians [44], which is analogous to the primary assumption made while formulating the particle filter. The Gaussian Particle Filter [46] and the Gaussian Sum Particle Filter [13] utilize this assumption by representing the prior in each iteration with a weighted sum of Gaussian densities. Each particle is then weighted according to the observation received, and updated according to the Extended Kalman Filter update algorithms. The posterior estimate is the weighted sum of all the particle posteriors. This avoids the problem of particle impoverishment.

The Gaussian Particle Filter also offers the following advantages, as derived in [46],
which make it well suited as the basis of a SLAM formulation:

1. Unlike the EKF, the assumption of additive Gaussian noise can be discarded for the GPF. Thus nonlinear motion models using non-additive non-Gaussian noises can also be implemented using the GPF.

2. The GPF tracks the mean and covariance of the state, which result in being a very computationally efficient algorithm, as shall be displayed further in this paper.

3. The GPF approximates the posterior distribution as Gaussians, due to which it is possible to discard resampling. This results in information being conserved in the process, which would have otherwise be lost by components as they were resampled. As noted in [40], discarding the resampling process also improves consistency of the vehicle pose in the FastSLAM algorithm as well.

4. Modifying the importance sampling procedure for the GPF, as mentioned in [46], makes it possible to propagate a higher moment than the covariance as well.
Chapter 3

Moving Window based Approaches to SLAM with Online Compensation of Changes in Motion Model Noise Covariance

3.1 Motivation

Standard Bayesian SLAM formulations assume perfect a priori knowledge of the vehicle proprioceptive sensor noise covariance, which may not be true in practice. Wrong proprioceptive sensor noise covariance may be the result of mismodelling, environmental factors, or due to the degradation of hardware as seen in fig3.1. In the event of unavailability or uncertainty of actual sensor covariance, the de facto rule is to use a large enough value as a conservative guess of the vehicle model covariance. A common assumption is that a larger covariance may just result in more computational time, while a smaller covariance results in a higher probability of estimator divergence.
Moving Window based Approaches to SLAM with Online Compensation of Changes in Motion Model Noise Covariance

Fig. 3.1 Hardware Degradation due to wear-and-tear from long term operation may result in change in covariance of vehicle noise statistics. An example is the degradation of the propeller of a marine vehicle due to extended operation in saline water.

Fig. 3.2 Comparison of RMS errors for different ratios of real and model standard deviations taken from [23]. The Y-axis measures RMS error, while the X-axis displays the ratio of modelled standard deviation to true standard deviation. The three different curves are characteristics for different number of state estimates, given by $n$. 
In practice, as seen in fig3.2, the net effect of a larger model covariance is similar to a smaller model covariance. In a study published in [23], Gelb et al performed a simulation using a one-dimensional Kalman filter. The model standard deviation was varied as a ratio of the true simulated standard deviation, and the resulting RMS error at the end of the simulation was recorded. As seen in fig3.2, the following findings can be inferred:

- **Minimum error:** As expected, Bayes optimality of the linear Kalman filter ensures that the RMS error is minimum when the model standard deviation equals the real standard deviation.

- **Mismodelled standard deviation:** The effect of a larger model covariance on the RMS error is the same as the effect of a smaller model covariance.

- **Result of number of estimates:** As expected, larger number of state estimates result better estimation, and consequently, smaller RMS error when the model and real standard deviations are equal. In other cases, the RMS error increases with increase in number of estimates. This result has serious implications for the formulation of SLAM algorithms, as it is commonly assumed that greater number of estimates will lead to lower RMS errors.

While the above findings are based on the linear Kalman filter, most Bayesian SLAM algorithms employ the Extended Kalman Filter in some form or the other, and hence can be assumed to inherit these characteristics from the Kalman Filter. Thus, a SLAM formulation which is able to compensate for changes in sensor covariance online will result in more accurate estimates.

This chapter will derive a Kalman Smoother based EKF-SLAM algorithm with an online compensation module for change in covariance. The derivation will be analyzed and its shortcomings will be discussed. An alternate approach employing the Adaptive Limited
Moving Window based Approaches to SLAM with Online Compensation of Changes in Motion Model Noise Covariance

Memory Filter (ALMF) will then be presented, and the results compared using simulation and experimental results.

3.2 The Kalman Smoother based EKF SLAM

3.2.1 Introduction to the Kalman Smoother

The Kalman Filter is an optimal estimator for linear stochastic systems affected by additive Gaussian noise, and can be summarised as the estimator of the expression:

\[ p(X_k|Y_k = y_0, y_1, ..., y_k) \]

Where \( X_k \) is the state estimate at time step \( k \), \( Y_k \) is the set of all measurements \( y_0, y_1, ..., y_k \) obtained till time step \( k \).

It consists of a time update (or prediction) step, and a measurement update step, and can be performed in real-time at the time of each observation to the system.

The Kalman Smoother algorithm is a complement of the Kalman Filter, and is used to post-process data to obtain a more accurate state estimate. It evaluates the expression:

\[ p(X_k|Y_N = y_o, ..., y_N) \]

Where \( k < N \).

The Combined Kalman Filter and Smoother Algorithm

The Kalman Smoother is deployed along with a Kalman Filter to perform a backward update and optimize the state estimate in post-processing, and hence cannot be deployed in real time. The combined algorithm is as follows:
Algorithm 1 Kalman Filter and Smoother algorithm Pseudocode

Kalman Filter Forward Update step:
for $k = 0; k < N; k + +$ do
  Generate prediction density:
  $\hat{x}_{k+1|k} = F\hat{x}_{k|k}$
  $P_{k+1|k} = FP_kF^T + Q$
  Calculate predicted observation:
  $\hat{y}_k = H\hat{x}_{k+1|k}$
  Input Observations: $y_k$
  Measurement Update:
  $K_{k+1} = P_{k+1|k}H^T(HP_{k+1|k}H^T + R^{-1})^{-1}$
  $\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}(y_{k+1} - \hat{y}_{k+1})$
  $P_{k+1|k+1} = P_{k+1|k} - K_{k+1}HP_{k+1|k}$
end

Kalman Smoother Backward Update step:
for $k = N - 1; k >= 0; k -$ do
  $L_k = P_{k|k}AT_kP_{k+1|k}$
  $\hat{x}_{k|N} = \hat{x}_{k|k} + L_k(\hat{x}_{k+1|N} - \hat{x}_{k+1|k})$
  $P_{k|N} = P_{k|k} + L_k(P_{k+1|N} - P_{k+1|k})L_k^T$
end

Parameter Estimation using the Kalman Smoother

The Kalman Smoother is also used to estimate parameters of the filter using an Expectation
Maximization (EM) algorithm. The expression for the optimal covariance thus obtained is
given by (further derivation in Appendix A):

$$Q_{EM} = \frac{1}{N-1} \sum_{k=0}^{N-1} (x_{k+1} - Fx_k)(x_{k+1} - Fx_k)'$$ (3.1)

Expression for Covariance of a Nonlinear Model

The vehicle motion model commonly used for localization and SLAM applications are non-
linear in nature, hence the above expression cannot be evaluated directly to derive the co-
variance matrix for the motion model.
Fig. 3.3 Robot Motion Model used. The turn angle is marked as G. The Wheelbase size is WB.

The expression for the prediction density is given by:

$$\hat{x}_{k+1|k} = f(\hat{x}_{k|k}, u_k)$$ (3.2)

Where $f$ is the transition model, and $u_k$ is the set of inputs at time $k$.

The resultant expression for the covariance matrix is as follows:

$$Q_{EM} = \frac{1}{N-1} \sum_{k=0}^{N-1} (x_{k+1|k+1} - f(x_{k|k}, u_k))$$ (3.3)

### 3.2.2 Simulation

A simulation was carried out to test the effectiveness of the Kalman Smoother-based SLAM algorithm. The motion model simulated is displayed in fig.3.1, given by eq3.4. A wheelbase
Fig. 3.4 Simulated Trajectory. The Trajectory taken by the vehicle shown in blue, the vehicle is represented as a red triangle. The real features are displayed as green asterisks. The estimated landmarks are shown as red points. The feature associations for the current scan are shown as yellow lines.

of 1.2m was used for the purpose of the simulation. The simulated trajectory is displayed in fig. 3.2.

\[
\begin{bmatrix}
\hat{x}_{k|k-1}(1) \\
\hat{x}_{k|k-1}(2) \\
\hat{x}_{k|k-1}(3)
\end{bmatrix}
=
\begin{bmatrix}
\hat{x}_{k-1|k-1}(1) \\
\hat{x}_{k-1|k-1}(2) \\
\hat{x}_{k-1|k-1}(3)
\end{bmatrix}
+
\begin{bmatrix}
u_k(1)dt \cdot \cos(\hat{x}_{k-1|k-1}(3) + u_k(2)) \\
u_k(1)dt \cdot \sin(\hat{x}_{k-1|k-1}(3) + u_k(2)) \\
\frac{u_k(1)dt}{W_B} \sin(u_k(2))
\end{bmatrix}
\]  

(3.4)

where control inputs \( u_k(1) \) is velocity input \( v_k \) and input \( u_k(2) \) is angle input \( G_k \).

Three cases were simulated as follows:

- Model covariance equals the real covariance

\[
Q_{\text{actual}} = Q_{\text{model}}
\]  

(3.5)
Fig. 3.5 RMS Error for case 1. $Q_{\text{actual}} = Q_{\text{model}}$

- Model covariance equals 1.3 times the real covariance

$$Q_{\text{actual}} = Q_{\text{model}}/1.3 \quad (3.6)$$

- Real covariance equals 1.3 times the model covariance

$$Q_{\text{actual}} = 1.3Q_{\text{model}} \quad (3.7)$$

30 Monte Carlo runs were carried out for each case, in order to average out the effects of outliers. The algorithm was benchmarked against the EKF-SLAM algorithm which was put through the same conditions.
3.2 The Kalman Smoother based EKF SLAM

3.2.3 Results

The Root Mean Square (RMS) errors for the pose-coordinates are plotted in figs. 3.3, 3.4 and 3.5. The error reduces at points in between the trajectory due to the observation of previously stored landmarks. The plots diverge towards the end of the trajectory, but the errors get constrained towards at the last one-sixth of the trajectory because all the observations are due to previously stored landmarks.

The Kalman Smoother-based EKF SLAM greatly outperforms the EKF-SLAM algorithm in all three cases. Furthermore, surprisingly, the performance of the Kalman Smoother-based algorithm shows remarkable consistency in all three cases. The error plots are almost similar in shape in all three cases, barring local variations.
Case 1: $Q_{actual} = Q_{model}$

As expected, the RMS errors are most similar in this scenario. The Kalman Smoother algorithm still outperforms the nonadaptive EKF-SLAM algorithm. Speculatively, this may be due to the Kalman Smoother decreasing the errors caused due to the nonlinearity of the Extended Kalman Filter.

Case 2: $Q_{actual} = Q_{model}/1.3$

The resultant error plots from this condition are similar to the error plots for the previous case. The Kalman Smoother-based EKF-SLAM algorithm, however, outperforms the nonadaptive EKF-SLAM algorithm by a larger degree in this scenario. Due to the model covariance being larger than the actual covariance in this case, the EKF-SLAM estimate is unable to correct the estimates quickly in this scenario.

Case 2: $Q_{actual} = 1.3Q_{model}$

The Kalman Smoother-based EKF-SLAM algorithm outperforms the nonadaptive EKF-SLAM algorithm by the largest extent in this scenario. Due to the model covariance being smaller than the actual covariance, the simulated noise in the motion model inputs does not lie within the predicted covariance estimate, causing the estimate to start diverging if left uncorrected.

3.2.4 Drawbacks of the Kalman Smoother

The Extended Kalman Filter attempts to convert nonlinear motion and observation models to linear Kalman Filter equivalents using Jacobians. As a result, the state estimate diverges from the ground truth as the nonlinearity of the motion model increases.

Since the Kalman Smoother-based algorithm is based on the Extended Kalman Filter, it
3.2 The Kalman Smoother based EKF SLAM

Fig. 3.7 RMS Error for case 3. $Q_{actual} = 1.3Q_{model}$

Fig. 3.8 RMS Error comparisons for three different rates of input. The inputs for the plot labelled 'Trajectory 2' is 10 times the inputs for the plot for 'Trajectory 1'. The inputs for the plot labelled 'Straight Line Path' is 0.1 times the inputs for the plot for 'Trajectory 1', making the path the most linear of the three scenarios.
was tested for similar vulnerabilities to nonlinearity. Three different scenarios were tested for rates of input classified by three cases shown below:

- **Standard Rate of input** $u_{std}$. Velocity is $1 \text{m/s}$, angular rate is $5^\circ/\text{s}$

- **Faster Rate of input** $u_{fast} = 10u_{std}$. The fast rate of input increases the nonlinearity of the change in estimates. The set of RMS errors obtained was upsamplied to fit the size of the RMS errors in the above case.

- **Slower Rate of input** $u_{slow} = 0.1u_{std}$. The slow rate of input makes the change in estimates more linear. The set of RMS errors obtained was downsampled to fit the size of the RMS errors in the above two cases.

The algorithm was run through the trajectory shown in fig.3.2 for 30 Monte Carlo runs for each of the scenarios.

Change in nonlinearity of the motion model results in a large change in the RMS errors of the simulated run. The more linear the path (or, alternatively, the smaller the change in inputs between time steps), the better the performance of the Kalman Smoother. Higher nonlinearity in the motion model leads to greater divergence of the linearised motion model from the true vehicle motion model. Further analysis of the effect of nonlinearity on vehicle transition model is performed in Chapter 5.

### 3.2.5 Conclusion

While initially seeming ideal as a model adaptation algorithm, the Kalman Smoother-based EKF-SLAM algorithm suffers from the inherent disadvantage of being derived from the Extended Kalman Filter. The subsequent search for adaptive state estimation algorithms displaying greater flexibility in formulation resulted in the Adaptive Limited Memory Filter (ALMF) algorithm, described in the next section.
3.3 Adaptive Limited Memory Filter for Nonlinear Models

The Adaptive Limited Memory Filter (ALMF) [53] was formulated for the standard Kalman Filter, and hence was affected by the issues arising from the assumptions used in its formulation. It was limited to linear process and noise models, which prohibited its use in most real world scenarios [51] dealing with nonlinear models, including vehicle localization and SLAM.

To overcome this disadvantage, the ALMF was reformulated with nonlinear models in [7] and implemented to the problem of vehicle localization with an unknown process noise covariance matrix, and an unknown time varying bias. This was then reformulated for SLAM and applied with two different kinds of SLAM methodologies.

The ALMF assumes the covariance noise is ergodic over \( N \) steps (\( N \) is a user defined input). The process noise covariance matrix is calculated for a window of 'N' steps, using the following formula:

\[
Q^* = \frac{1}{N} \sum_{i=k-N}^{k} F_i^{-1} [ \delta x_i \delta x_i^T - (P_i - P_{i-1} - F_i \hat{Q}_{i-1} F_i^T) (F_i^{-1})^T ]
\]  

where \( \delta x_k \) is the average residual calculated over the window of 'N' steps.

A weighted sum is then carried out with the previous covariance matrix, and the window covariance matrix, multiplied with appropriate weights representing the confidence of the researcher. This weighted sum is used as the new covariance matrix for the next window of \( N \) time steps.
Algorithm 2 Adaptive Limited Memory Filter for Nonlinear Models

for all $x_k$ do
    predict $x_{k|k-1}$
    calculate predicted observations $\hat{z}_k$
    obtain real observations $z_k$
    calculate updated estimate $x_{k|k}$
    calculate residual $\delta x_k = z_k - \hat{z}_k$
    calculate step covariance $Q_k$
    if $k$ is a multiple of window size $N$ then
        Calculate window covariance $Q_{k/N}^*$
        Compute new process model covariance as weighted sum of $Q_k$ and $Q_{k/N}^*$
    end
end

3.4 The Adaptive Extended Kalman Filter Simultaneous Localization and Mapping (AEKF-SLAM) Algorithm

3.4.1 Formulation

The Adaptive Extended Kalman Filter SLAM (AEKF-SLAM) algorithm was proposed and implemented using the ALMF algorithm as detailed above. The weights used to calculate the new covariance matrix were the number of steps used to calculate the respective covariance matrices.

$$\hat{Q}_{k} = \frac{NQ^* + (N_{\text{exp}} + (k-1)N - N)\hat{Q}_{k-1}}{N_{\text{exp}} + kN}$$ (3.9)

Where $k$ is the window number, and $N_{\text{exp}}$ is the expected window number of inputs used to estimate the initial covariance matrix.
Algorithm 3 AEKF-SLAM algorithm

Initialization $(x_0|0, P_0|0, Q_0, N_{exp})$

for all $x_k$ do

predict

$x_{k|k-1} = f(x_{k|k-1}, u_k, v_k)$

$F_k = \frac{\partial f(x, u)}{\partial x}|_{x_{k|k-1}, u_k}$

$M_k = \frac{\partial f(x, u)}{\partial v}|_{x_{k|k-1}, u_k}$

$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + M_k Q_k M_k^T$

predicted observations $\hat{z}_k = h_k(x_{k|k-1}, w_k)$

obtain real observations $z_k$

update residual $\delta x_k = z_k - \hat{z}_k$

$H_k = \frac{\partial h_k(x)}{\partial x}|_{x_{k|k-1}}$

$L_k = \frac{\partial h_k(x)}{\partial w}|_{x_{k|k-1}}$

residual covariance $S_k = H_k P_{k|k-1} H_k^T + L_k R_k L_k^T$

Kalman Gain $K_k = P_{k|k-1} H_k^T S_k^{-1}$

Update estimate $x_{k|k} = x_{k|k-1} + K_k \delta x_k$

Update Covariance $P_{k|k} = (I - K_k H_k) P_{k|k-1}$

Step covariance $Q_k^* = F_k^{-1} [\delta x_k \delta x_k^T - (P_{k|k} - P_{k-1|k-1} - F_k \hat{Q}_{k-1} F_k^T)(F_k^{-1})^T$

Update process model covariance

if $k$ is a multiple of window size $N$ then

calculate window covariance $Q_{k/N}^* = \frac{1}{N} \sum_{i=k-N}^{k} Q_i^*$

New process model covariance $\hat{Q}_k = \frac{N Q_{k/N}^* + (N_{exp} + (k-1)N - N) \hat{Q}_{k-1}}{N_{exp} + kN}$

end
Moving Window based Approaches to SLAM with Online Compensation of Changes in Motion Model Noise Covariance

Fig. 3.9 RMS Error for case 1. $Q_{\text{actual}} = Q_{\text{model}}$

Fig. 3.10 RMS Error for case 2. $Q_{\text{actual}} = Q_{\text{model}}/1.3$
### 3.4.2 Simulation Results

The algorithm was benchmarked against the nonadaptive EKF-SLAM algorithm, as well as the Kalman Smoother-based EKF SLAM algorithm through the simulation methodology described in chapter 3.2.

The results are plotted in figs. 3.9, 3.10, 3.11. As seen in the figures, the AEKF-SLAM algorithm clearly overperforms both the EKF-SLAM and the Kalman Smoother-based EKF SLAM algorithm.

### 3.4.3 Experimental Results

The necessity of performing adaptive SLAM in the presence of GPS localization in a noisy environment with observations suffering from clutter was analyzed. Data was collected with the help of the Center for ENvironmental Sensing And Modelling (CENSAM) in the littoral waters off the coast of Singapore in the Selat Puah marine environment.
Moving Window based Approaches to SLAM with Online Compensation of Changes in Motion Model Noise Covariance

Fig. 3.12 Experimental Setup used to gather Data

Fig. 3.13 Sample Scan obtained, overlayed on Google Earth image for reference.
3.4 The Adaptive Extended Kalman Filter Simultaneous Localization and Mapping (AEKF-SLAM) Algorithm

Fig. 3.12 displays the hardware setup employed to obtain data. The Autonomous Surface Craft used was a robotic sea-kayak which represents a low cost, high load bearing platform, being highly maneuverable and capable of operating in shallow waters. For stabilisation in the choppy waters common to Singapore’s Selat Puah, lateral buoyancy aids were added to the platform, as depicted in the figure. The ASC was equipped with a GPS receiver and low cost DSP5000 single-axis gyroscope for 3D pose measurements at each time $k$ and was drivable / steerable via a rear mounted remote control electric thruster.

Fig. 3.13 shows a sample scan of the data obtained, overlayed with a Google Earth picture for geographic context. Exteroceptive observations were obtained using an off-the-shelf XBand marine radar (depicted as orange-green ‘haze’), while proprioceptive readings were taken from a single-exis gyroscope (orientation shown as a red line). Automatic Identification System (AIS) readings were added in as ground truth of nearby ships and marine platforms (seen as green dots). The GPS coordinates of nearby buoys and lighthouses were manually entered (shown as black dots).

Clutter analysis of the scans were performed, and overall detection probability of the features estimated. To extract features, a 2D Gaussian low pass filter was convolved with the regions of the scan identified by the adaptive threshold obtained by analysis of the clutter affecting each scan. The smoothed features were then clustered based on nearby pixels and pruned according to a minimum and maximum area constraints.

The trajectories obtained via EKF SLAM and AEKF SLAM are shown in figs. 3.14 and 3.15 respectively for comparison. The ground truth obtained via GPS is in green, and the estimated trajectory is in blue. As seen in the figures, not only does the AEKF-SLAM vastly outperform the EKF SLAM in the trajectory estimate, being able to close the loop, but the map obtained via the AEKF SLAM is much sparser, since it contains fewer false associations, leading to a better trajectory estimate as well.

Fig. 3.16 compares the Mean Square Errors obtained from the EKF SLAM and the
AEKF SLAM implementations. Between time step 1250 and 1300, the trajectory of the ASC undergoes a sharp turn, leading to many observations being associated wrongly, causing a sharp rise in error in the trajectory estimate, which is reflected in the MSE graph. The AEKF SLAM, however, is able to estimate the change in motion model statistics, and performs better.

3.5 Conclusion

This chapter explored the formulation of an EKF-SLAM algorithm with a compensation module for changes to proprioceptive sensor covariance. An initial derivation using the Kalman Smoother was presented and its shortcomings analysed. The Adaptive Limited Memory Filter (ALMF) was then explored as a possible alternative, and the Adaptive Ex-
The extended Kalman Filter SLAM (AEKF-SLAM) algorithm was derived. The AEKF-SLAM algorithm compared favorably with the Kalman Smoother in simulation, and hence was deployed on data obtained from a marine dataset.

The positive results from the AEKF-SLAM algorithm suggests a viable approach for formulating an adaptive sensor fusion approach to SLAM. The next chapter will attempt to extend this formulation to compensate for bias in conjunction with the particle filter based FastSLAM algorithm. The drawbacks of the ALMF algorithm inherent in the formulation will then be explored and explained in greater detail.
Fig. 3.16 Comparison of AEKF SLAM and EKF SLAM MS Errors (Blue: MS Error due to EKF-SLAM, Red: MS Error due to AEKF-SLAM)
Chapter 4

Extension of Moving Window Approach to Bias Compensation and Drawbacks Encountered

4.1 Motivation

Most Bayesian SLAM approaches assume the proprioceptive sensor to be affected by noise exhibiting no bias. This assumption does not hold true in autonomous platforms which operate in changing environmental conditions as seen in fig. 4.1. Some examples of envi-

Fig. 4.1 Environmental causes for bias may be unmodelled gusts of wind (left), or change in expected odometry slip due to navigation along an incline (right)
Environmental bias include:

- Quadcopters affected by gusts of wind.

- Marine vehicles affected by sea currents or the wake of passing boats.

- Change in wheel slip due to inclines or movement of the platform between wet and dry surfaces.

- Change in wheel slip due to change in vehicle load, such as the alighting of passengers from an autonomous vehicle.

Previous attempts at bias compensation [59], [69], [58] are based on the Extended Kalman Filter, and are internally focussed by assuming the sensor drift to be the cause of the bias. Consequently, all formulations attempt to estimate the sensor drift by extending the Extended Kalman Filter state to include bias variables. This approach is unsuitable to rapidly changing environmental biases. A SLAM formulation which is able to compensate for temporary environmental biases will, thus, be able to produce a more accurate estimate.

This chapter extends the Adaptive Limited Memory Filter (ALMF) based approach taken in the last chapter, to simultaneously compensate for unmodelled environmental bias, as well as change in covariance of the proprioceptive sensor. The formulation is implemented in tandem with the particle filter based FastSLAM [28], [45], [55], [46] to lead to the AEKF-FastSLAM algorithm [61]. The algorithm will be favorably compared with previous state of the art approaches using simulated and experimental data.

This chapter will then discuss the drawbacks of all three moving window algorithms discussed thus far by analysing the results of extensive simulations.
4.2 ALMF with Bias Estimation - The AEKF-FastSLAM Algorithm

The ALMF algorithm was also used in formulating and implementing an Adaptive Kalman Filter approach to the Factorised Solution to Simultaneous Localization and Mapping (FastSLAM) [61]. While similar to the algorithm employed in the AEKF-SLAM algorithm, there was a bias-estimation component added in to deal with local biases in the marine environment, due to factors such as sea currents and the wake of passing boats.

Algorithm 4 AEKF-FastSLAM algorithm with bias estimation

Initialization \((x_0|0, P_0|0, Q_0, N_{\text{exp}})\)

for all \(x_k\) do
  predict
  \[ x_{k|k-1} = f(x_{k|k-1}, u_k, v_k) \]
  \[ F_k = \frac{\partial f(x, u)}{\partial x} |_{x_{k|k-1}, u_k} \]
  \[ M_k = \frac{\partial f(x, u)}{\partial v} |_{x_{k|k-1}, u_k} \]
  \[ P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + M_k Q_k M_k^T \]
  predicted observations \(\hat{z}_k = h_k(x_{k|k-1}, w_k)\)
  obtain real observations \(z_k\)
  update
  residual \(\delta x_k = z_k - \hat{z}_k\)
  \[ H_k = \frac{\partial h_k(x)}{\partial x} |_{x_{k|k-1}} \]
  \[ L_k = \frac{\partial h_k(x)}{\partial w} |_{x_{k|k-1}} \]
  residual covariance \(S_k = H_k P_{k|k-1} H_k^T + L_k R_k L_k^T\)
  Kalman Gain \(K_k = P_{k|k-1} H_k^T S_k^{-1}\)
  Update estimate \(x_{k|k} = x_{k|k-1} + K_k \delta x_k\)
  Update Covariance \(P_{k|k} = (I - K_k H_k) P_{k|k-1}\)
  Step covariance \(Q^*_{k} = F_k^{-1} (\delta x_k \delta x_k^T - (P_{k|k} - P_{k-1|k-1} - F_k \hat{Q}_{k-1} F_k^T)(F_k^{-1})^T)\)
  Update process model covariance
  if \(k\) is a multiple of window size \(N\) then
    calculate window covariance \(Q_{k/N}^* = \frac{1}{N} \sum_{i=k-N}^{k} Q_i^*\)
    New process model covariance \(\hat{Q}_k = \frac{N Q_{k/N}^* + (N_{\text{exp}} + (k-N-N) \hat{Q}_{k-1})}{N_{\text{exp}} + kN}\)
  end
end
4.2.1 Formulation

The bias is assumed ergodic over $N$ steps. First, a step bias was calculated at each step:

$$ q_k = x_{k|k} - x_{k|k-1} + \hat{b}_{k-1} $$

(4.1)

where $k_s$ is the current window number.

After $N$ steps, a window bias was calculated by averaging the individual step biases.

$$ \hat{b}_{k_s} = \frac{1}{N} \sum_{i=k-N+1}^{k} q_i $$

(4.2)

The covariance matrix for the window bias is then calculated.

$$ \hat{B}_{k_s} = \frac{1}{N-1} \sum_{i=k-N+1}^{k} (q_i - \hat{b}_{k_s}) (q_i - \hat{b}_{k_s})^T $$

(4.3)

The covariance matrix of all the particles is then calculated using the formula from the Adaptive Limited Memory Filter given by:

$$ \hat{W} = \frac{1}{N_{\text{PART}}} F_k^{-1} \left[ \delta_{x_k} \delta_{x_k}^T - (P_k - P_{k-1}) - F_k \hat{Q}_{k_{s-1}} F_k^T (F_k^{-1})^T \right] $$

(4.4)

This is then added to the bias covariance to obtain the window covariance matrix.

$$ \hat{Q}_{k_s} = \hat{B} + \hat{W} $$

(4.5)
Again, like the ALMF, a weighted average is performed between the window covariance matrix, and the previous covariance matrix, to obtain the new covariance matrix, to be used for calculations and sampling for the next window. The weights used must reflect the confidence of the researcher, and hence, the number of time steps used to estimate the covariance matrix is considered.

\[
\hat{Q}_k = \frac{NQ^* + (N_{exp} + (k-1) \ast N - N)\hat{Q}_{k-1}}{N_{exp} + k \ast N}
\] (4.6)

4.2.2 Experimental Results

This algorithm was again evaluated using the data obtained from sea trials done with CEN-SAM, using the hardware setup as shown in fig.4.4. The results were processed through the FastSLAM algorithm, and compared with the AEKF-FastSLAM algorithm.

Figs.4.9, 4.10, 4.11 display the trajectory estimates obtained via standard FastSLAM, AEKF-FastSLAM without bias estimation, and AEKF-FastSLAM with the bias estimation component, respectively. The Mean Square Error comparison graph is shown in Fig.4.12, in which the MSE due to standard FastSLAM (in magenta), EKF-SLAM(green), ALMF-based FastSLAM without bias estimation (red) and ALMF based FastSLAM with bias estimation (blue) are compared.

4.3 Drawbacks of the Moving Window Approach

The AEKF algorithm, while able to adapt to external changing unmodeled noise, is greatly dependant on the window size, a heuristic determined by the researcher.

Thus, the actual duration of the effect of the unmodeled noise may not match with the window size provided by the researcher, which may lead to further divergence of the esti-
This hypothesis was tested using a simulated route affected by unmodeled noise at two locations. An Ackermann process model was used for this simulation. The subsequent sections will describe the results of the simulations, and provide a deeper analysis of the results.

### 4.3.1 Simulation

A simulation run was carried out as shown in fig.4.6 with a dense feature-based map environment. The simulation was carried out for five cases:

- With no unmodeled drift
- With fixed unmodeled drift
- With unmodeled drift displaying normal distribution characteristics
For the unmodeled drift scenarios, the drift was constrained to a fraction of the ground truth control inputs given by:

\[ \delta v_k = 0.3 \cdot v_k \] \hspace{1cm} (4.7a)

\[ \delta \theta_k = 0.3 \cdot \theta_k \] \hspace{1cm} (4.7b)

where

\[ \delta v_k \] is the unmodeled velocity drift.

\[ \delta \theta_k \] is the unmodeled angular drift.
Fig. 4.4 Trajectory using AEKF-FastSLAM with bias estimation (Green: Ground truth, Blue: AEKF-FastSLAM estimate, Blue crosses: Estimated landmarks, Yellow lines: Superimposed observations, Red lines: Data Association)

$v_k$ is the velocity ground truth

$\theta_k$ is the angular ground truth

The unmodeled drift in the random drift scenario was modeled as a normal distribution given by:

$$\sigma_{\delta v_k} = 0.3 \cdot \sigma_{v_k} \quad (4.8a)$$

$$\sigma_{\delta \theta_k} = 0.3 \cdot \sigma_{\theta_k} \quad (4.8b)$$

where

$\sigma_{\delta v_k}$ is the unmodeled velocity drift standard deviation.

$\sigma_{\delta \theta_k}$ is the unmodeled angular drift standard deviation.

$\sigma_{v_k}$ is the velocity ground truth standard deviation.
4.3 Drawbacks of the Moving Window Approach

Fig. 4.5 Mean Square Error Comparison (Magenta: FastSLAM RMS error, Red: RMS Error of AEKF-FastSLAM without bias estimation, Blue: RMS Error of AEKF-FastSLAM with bias estimation, Blue: RMS Error of EKF-SLAM)

\( \sigma_{\theta_k} \) is the angular ground truth standard deviation

In the unmodeled drift scenarios, an additive unmodeled drift was simulated in the green portion of the trajectory (steps 120-240), while a subtractive unmodeled drift was simulated in the red portion of the trajectory (steps 440-560).

Scenarios involving mismatched model covariances are simulated as follows.

For the scenario simulating the model covariance smaller than the actual vehicle covariance, the model covariance is simulated as follows:

\[
\sigma_{v,\text{model}} = (\sigma_{v,\text{actual}})/1.3
\]  
(4.9a)

\[
\sigma_{\theta,\text{model}} = (\sigma_{\theta,\text{actual}})/1.3
\]  
(4.9b)
Fig. 4.6 Simulated trajectory of the platform. The red and green trajectory lines depict the presence of simulated bias for simulations analyzing the effects of bias. The green line represents the inclusion of additive bias (simulating motion downslope or in the direction of wind), while the red line represents subtractive bias (simulating motion upslope or against the direction of wind).

Similarly, the model covariances of the scenario simulating the model covariance larger than the actual vehicle covariance are calculated as follows:

\[
\sigma_{v,\text{model}} = (\sigma_{v,\text{actual}}) \times 1.3 \tag{4.10a}
\]

\[
\sigma_{\theta,\text{model}} = (\sigma_{\theta,\text{actual}}) \times 1.3 \tag{4.10b}
\]

where \(\sigma_{v,\text{actual}}\) and \(\sigma_{\theta,\text{actual}}\) are the standard deviations of the ground truth vehicle, and \(\sigma_{v,\text{model}}\) and \(\sigma_{\theta,\text{model}}\) are the standard deviations of the process model.

A scan rate of 10 Hz was simulated for the exteroceptive sensor, while a sensor output rate of 50 Hz was assumed for the proprioceptive sensor. The SLAM update was carried out after each exteroceptive reading, while the Adaptive Limited Memory Filter (ALMF)
Fig. 4.7 Estimation without noise

update took place after the specified number of updates given by the window size.

Each of these scenarios was performed for 50 Monte Carlo runs for three ALMF window sizes:

- Small window size of 5 steps per window, ALMF update rate 2 Hz.
- Medium window size of 25 steps per window, ALMF update rate 0.4 Hz.
- Large window size of 75 steps per window, ALMF update rate 0.133 Hz.

### 4.3.2 Results

This section analyzes the results of 50 Monte Carlo runs.

**No Unmodeled drift and Model Covariance Equal to the Vehicle Covariance**

This scenario is the default assumption for all Simultaneous Localization and Mapping algorithms. The noise statistics of the process model match the noise statistics of the ground truth vehicle. The Root Mean Square (RMS) errors for this scenario are displayed in fig.4.7.
Extension of Moving Window Approach to Bias Compensation and Drawbacks

Encountered

Being designed for precisely this scenario, the FastSLAM algorithm outperforms the ALMF-based algorithms. Among the ALMF-based algorithms, the filters based on the medium window size, as well as the small window size perform similarly, while the large window size based filter performs poorer than the others.

Fixed Unmodeled Drift

A fixed unmodeled drift was provided to the control inputs of the process model, as described earlier.

As seen in the fig.4.8, the RMS error of the FastSLAM algorithm spikes at the points of application of the drift (steps 120-240 for additive drift, and steps 440-560 for subtractive drift). As the loop is closed, the error decreases due to the observation of previously recorded features.

The ALMF algorithms display a similar pattern as well, with smaller spikes at the points of application of the drift, with lesser amplitude due to the adaptivity of the algorithms. Once again, the large window ALMF algorithm displays poorer performance than the ALMF algorithms with small and medium sized windows.
4.3 Drawbacks of the Moving Window Approach

Random Unmodeled Drift

The unmodeled drift affecting the control inputs of the process model was simulated with normal distribution statistics in this scenario.

The characteristics of the RMS error were similar to those of the RMS error for the scenario with fixed drift described in the previous subsection, as seen in fig.4.9. The medium window sized filter outperforms all the other filters for a major portion of the simulation run, while the small window sized filter performs at par with the FastSLAM algorithm.

The performance of the large window size filter, however, is the poorest of all the scenarios, and worse than the previous scenario with a fixed drift.

Model Covariance Larger than Vehicle Ground Truth Covariance

The model covariance is larger than the vehicle ground truth covariance in this scenario.

In cases of poorly calculated vehicle noise covariances, this approximation is often employed, wherein the model covariance is intentionally kept larger than the estimated vehicle model covariance.
In this scenario, as seen in fig.4.10 the RMS errors due to FastSLAM and the small and medium sized ALMF algorithms perform at par. The large window sized ALMF algorithm, however, performs worse than the other algorithms by a large margin.

Model Covariance Smaller than Vehicle Ground Truth Covariance

In order to maintain an acceptable convergence rate of the SLAM algorithm estimate to the ground truth, the model covariance is desired to be as close to the vehicle ground truth as possible. In cases of mismodeling, however, the model covariance is smaller than the ground truth covariance, leading to a smaller spread of FastSLAM particle hypotheses, resulting in divergence of the estimate.

As seen in the RMS error characteristics shown in fig.4.11, the FastSLAM algorithm and the small-sized ALMF algorithm perform at par for this scenario. The medium-sized window, however, performs better than the other three algorithms. The large window sized ALMF algorithm performs the worst among the four algorithms.
Fig. 4.11 Estimation with model covariance smaller than actual covariance

4.4 Conclusion

In most scenarios described above, the large window sized ALMF algorithm performed the worst of all four algorithms. The reason of this underperformance is due to the low frequency of noise statistic recalculation, resulting in the filter overcompensating in a window for unmodeled noise encountered in the previous window.

In the case of the small sized window filter, the algorithm performed worse, or at par with the nonadaptive FastSLAM algorithm. The reason for its poor performance is the rapid recalculation of the vehicle noise statistics, leading to poor convergence rate of the SLAM algorithm estimate.

The medium sized window filter uniformly outperforms all other algorithms considered in all cases involving mismodeled noise statistics or external unmodeled noise.

However, the algorithm employs a medium sized filter only in relation to the external noise being simulated in the scenarios considered. In other scenarios involving sudden random drifts due to wind gusts at one extreme, or constant drift due to sensor error on the other extreme, this window size may be considered large or small, respectively. Hence, it is
impossible to determine the right window size a-priori without knowledge of the potential disturbances that may affect the vehicle during its operation.

This drawback exists for all algorithms dependant on a memory window for adaptivity, such as the Adaptive Limited Memory Filter, or algorithms based on Kalman Smoothing. Hence, an alternative procedure involving estimation of noise parameters at every time iteration of the vehicle operation will be considered in subsequent chapters.
Chapter 5

A Comparison of SLAM Prediction Densities using the Kolmogorov Smirnov Statistic

5.1 Introduction

Pose and trajectory estimation modules performing localization or Simultaneous Localization and Mapping (SLAM) are essential components of any robot navigation system. A sensor fusion algorithm, which probabilistically fuses observations from two or more sensors via vehicle transition and observation models, is used to estimate pose and trajectory.

In order to ensure faster correction of the estimate by incoming observations, the algorithms must have a prediction density as close to the ground truth as possible. In order to ensure this, a greater portion of the algorithm prediction density must coincide with the vehicle prediction density.

Current state of the art metrics for comparison of SLAM algorithms include comparison of estimates using Root Mean Square Error, convergence analysis, and consistency analy-
A Comparison of SLAM Prediction Densities using the Kolmogorov Smirnov Statistic

Fig. 5.1 Hardware Setup. The robot platform contains sonars, panoramic and stereo cameras, a GPS sensor, one 2D planar LiDAR and one back-and-forth LiDAR.

sis, using previously collected standard datasets. If the SLAM algorithms being compared use the same sensors and similar strategies for map management and feature extraction, the main differentiator between the algorithms becomes their method of generating prediction densities. The drawback of comparing the estimates at the end of the prediction-correction cycle at each step is that when a highly accurate sensor is used to correct the prediction, comparison of the subsequent a-posteriori errors do not reveal the true picture of the differences between the algorithms. Conversely, a metric that compares only prediction densities of the SLAM algorithms is a sufficient measure of the accuracy of the algorithms.

This chapter will compare these schema for propagating the prediction densities, with the use of the Kolmogorov-Smirnov (KS) statistic[23] as a metric of comparison. The KS-statistic is a non-parametric goodness-of-fit test statistic which compares a continuous one-dimensional probability density with a sample set. Recent modifications allow the comparison of multidimensional data sets[33].

The Monte Carlo density asymptotically converges to the true prediction density, hence
the particle filter prediction density using an appropriately large number of particles (in our case, 5000 particles), is assumed to be equivalent to the true prediction density.

The particle filter propagates multiple hypotheses, each from a unique random sample generated from the noise statistics of the sensor. Thus, each particle can be considered an independent observation of the prediction density. This chapter will use this assumption to compare each prediction density generated by varying input noise with a set of particles generated from equivalent noise samples.

An alternate methodology employing the simulation of just the ground truth limits the comparison to only the mean of the prediction densities. The schema employed in this chapter will enable the comparison of the prediction densities upto the second order (mean and covariance), which would greatly improve the analysis of the algorithms.

The structure of this chapter is as follows. Section 5.2 discusses the formulation of the predicted densities for the algorithms being compared and the expression of the Kolmogorov-Smirnov test statistic. Section 5.3 describes the implementation of the comparison scheme on the Ackermann motion model. Section 5.4 analyzes the results obtained from the simulations. Section 5.5 describes an experiment performed to validate the analysis using real world data, and Section 5.6 concludes the analysis.

5.2 Formulation

The equations governing the generation of the prediction density using each algorithm (particle filter, GPF, EKF, UKF and CDKF) are discussed in this section.

5.2.1 Motion Model Noise Statistics

The most recent estimate obtained is assumed to be $\hat{x}_{k-1|k-1}$ at iteration $k - 1$. The prediction density at step $k$ (viz, $x_{k|k-1}$) is to be generated. A control noise of $u_k$ affected by noise $v_k$ is
A Comparison of SLAM Prediction Densities using the Kolmogorov Smirnov Statistic

the input. The nonlinear prediction generation function is \( f(x_{k-1}, v_k, u_k) \).

where,

\[ v_k \sim \mathcal{N}(0, Q_k) \]  

(5.1a)

\[ x_{k|k-1}, x_{k-1|k-1} \epsilon \mathbb{R}^n \]  

(5.1b)

\[ f : \mathbb{R}^n \rightarrow \mathbb{R}^n \]  

(5.1c)

\[ u_k, v_k \epsilon \mathbb{R}^r \]  

(5.1d)

5.2.2 Prediction Density Approximation

The particle filter generates a prediction density of \( N_{PF} \) independent hypotheses using an equal number of noise samples generated from the motion model noise density. Each particle \( i \) is generated according to the following equation.

\[ x_{PF,k}^i = f(x_{PF,k-1|k-1}^i, v_k^i, u_k) \]  

(5.2)

The particle filter prediction density is a weighted sum of all the particles, given by the following expression.

\[ x_{PF,k|k-1} = \sum_{i=1}^{N_{PF}} w_{PF,k-1}^i x_{PF,k}^i \delta(x_k^i) \]  

(5.3)

The asymptotic property of the particle filter states that as \( N \rightarrow \infty \), \( x_{PF,k|k-1} \rightarrow x_{TRUE} \), the true prediction density.

The Gaussian Particle Filter (GPF) approximates the prediction density as a sum of
Fig. 5.2 Simulated Trajectory. Axes dimensions are in metres. The robot is represented by a red triangle, and its trajectory is displayed by red *’s. The particle filter hypotheses are displayed as cyan dots. Covariances are displayed as ellipses, while estimates are displayed as +s. EKF predictions are in blue, GPF in black, UKF in green and CDKF in yellow.
$N_{GPF}$ normal distributions with the mean of each distribution generated using random noise samples from the noise covariance matrix. Each distribution $i$ is generated according to the following expression:

$$x_{GPF,k}^i \sim \mathcal{N}(\hat{x}_{GPF,k}^i, P_{GPF,k}^i)$$  \hspace{1cm} (5.4)

where

$$\hat{x}_{GPF,k}^i = f(x_{GPF,k-1|k-1}^i, v_k^i, u_k)$$  \hspace{1cm} (5.5)

$$P_{GPF,k}^i = \frac{1}{N_{GPF}}$$  \hspace{1cm} (5.6)

The prediction density is

$$x_{GPF,k|k-1} = \sum_{i=1}^{N_{GPF}} w_{GPF,k-1}^i x_{GPF,k}^i$$  \hspace{1cm} (5.7)

The Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) and Central Difference Kalman Filter (CDKF) represent the prediction density as a normal distribution given by the following expressions.

$$x_{EKF,k} \sim \mathcal{N}(\hat{x}_{EKF,k}, P_{EKF,k})$$  \hspace{1cm} (5.8)

$$x_{UKF,k} \sim \mathcal{N}(\hat{x}_{UKF,k}, P_{UKF,k})$$  \hspace{1cm} (5.9)

$$x_{CDKF,k} \sim \mathcal{N}(\hat{x}_{CDKF,k}, P_{CDKF,k})$$  \hspace{1cm} (5.10)

### 5.2.3 Prediction Density Generation Equations

The expressions for the densities of the EKF, UKF and CDKF are calculated as follows.
5.2 Formulation

Extended Kalman Filter Prediction Density

The EKF prediction is given by:

\[ x_{EKF,k|k-1} = f(\hat{x}_{EKF,k-1|k-1}, v_k, u_k) \]  \hspace{1cm} (5.11)

In order to calculate the covariance matrix, the process model Jacobians are first calculated as:

\[ F_{x,k} = \nabla_x f(x, v_k, u_k) \bigg|_{x=\hat{x}_{k|k-1}} \]  \hspace{1cm} (5.12)

\[ F_{w,k} = \nabla_x f(\hat{x}_{EKF,k-1|k-1}, v, u_k) \bigg|_{v=v_k} \]  \hspace{1cm} (5.13)

The process model covariance matrix is calculated using the Jacobians using the following expression:

\[ P_{EKF,k|k-1} = F_{x,k} P_{x,EKF,k|k-1} F_{x,k}^T + F_{w,k} Q_k F_{w,k}^T \]  \hspace{1cm} (5.14)

Unscented Kalman Filter Prediction Density

Prior to calculating the Unscented Kalman Filter prediction density, the sigma-points are calculated as follows:

\[ \chi_{k-1}^\alpha = \begin{bmatrix} \hat{x}_{UKF,k-1|k-1} \\ \hat{x}_{UKF,k-1|k-1} + \gamma \sqrt{P_{UKF,k-1|k-1}} \\ \hat{x}_{UKF,k-1|k-1} - \gamma \sqrt{P_{UKF,k-1|k-1}} \end{bmatrix}^T \]  \hspace{1cm} (5.15)
A Comparison of SLAM Prediction Densities using the Kolmogorov Smirnov Statistic

where

\( \hat{x}_{UKF,k-1|k-1}^a \) is the augmented state matrix given by:

\[
\hat{x}_{UKF,k-1|k-1}^a = \begin{bmatrix}
\hat{x}_{UKF,k-1|k-1}^T \\
w_k^T
\end{bmatrix}
\]  \hspace{1cm} (5.16)

\( P_{UKF,k-1|k-1}^a \) is the augmented covariance matrix with the initial value given by

\[
P_{UKF,0}^a = \begin{bmatrix}
P_{UKF,x0} & 0 \\
0 & Q_k
\end{bmatrix}
\]  \hspace{1cm} (5.17)

and \( \gamma \) is a composite scaling parameter given by \( \gamma = \sqrt{L_{UKF} + \lambda} \), where \( L_{UKF} \) is the dimension of \( \hat{x}_{k-1|k-1}^a \) and the method to calculate \( \lambda \) is given in [1]. The UKF prediction density is calculated as follows:

\[
\chi_{x,k|k-1} = f(\chi_{x,k-1|k-1}, \chi_{w,k-1|k-1}, u_k)
\]  \hspace{1cm} (5.18)

\[
\chi_{UKF,k|k-1} = \sum_{i=0}^{2L_{UKF}} w^{(m)}_i \chi_{i,k|k-1}^x
\]  \hspace{1cm} (5.19)

\[
P_{UKF,k|k-1} = \sum_{i=0}^{2L_{UKF}} w^{(c)}_i (\chi_{i,k|k-1}^x - \hat{x}_{UKF,k|k-1})(\chi_{i,k|k-1}^x - \hat{x}_{UKF,k|k-1})^T
\]  \hspace{1cm} (5.20)

where

the expressions for \( L_{UKF}, \chi_{x,k|k-1}^x \), the weights \( w^{(m)}_i \) and \( w^{(c)}_i \) are provided in [1].
Fig. 5.3 The particle ordering scheme for the calculation of the Kolmogorov-Smirnov statistic is explained in this figure. The trajectory is displayed with cyan asterisks, the current pose is in blue colour and the generated particles are in red colour. 5 particles are simulated for illustration purposes, with their ordering displayed in text alongside the respective particles. The particles are labeled in order of signed angle change. In case of two particles with the same angle change, the particle velocities are used as tie-breaker.

Central Difference Kalman Filter Prediction Density

For the CDKF, the covariance square-root column vectors are first computed as follows:

\[ s^{x,i}_{CDKF,k-1} = h(\sqrt{P_{x,k-1|k-1}}) ; i = 1, \ldots, L_x \]  (5.21)

\[ s^{w,i}_{CDKF,k-1} = h(\sqrt{Q_k}) ; i = 1, \ldots, L_v \]  (5.22)

where

\( L_x \) and \( L_v \) are the dimensions of the state and process noise random variable.

\( h \) is the Central Difference interval size. [1] suggests \( h = \sqrt{3} \) as the optimal value for Gaussian random variables.
A Comparison of SLAM Prediction Densities using the Kolmogorov Smirnov Statistic

These column vectors are then used to calculate the state and covariance equations.

\[ x_{\text{CDKF}, k|k-1} = \frac{h^2 - L_x - L_w}{h^2} f(\hat{x}_{k-1|k-1}, v_k, u_k) \]

\[ + \frac{1}{2h^2} \sum_{i=1}^{L_x} \left[ f(\hat{x}_{k-1|k-1} + s_{k-1}^{x,i}, v_k, u_k) \right. \]

\[ + f(\hat{x}_{k-1|k-1} - s_{k-1}^{x,i}, v_k, u_k) \]

\[ + \frac{1}{2h^2} \sum_{i=1}^{L_x} \left[ f(\hat{x}_{k-1|k-1} + s_{k-1}^{v,i}, v_k + u_k) \right. \]

\[ + f(\hat{x}_{k-1|k-1}, v_k - s_{k-1}^{v,i}, u_k) \] (5.23)

\[ P_{\text{CDKF}, k|k-1} = \]

\[ \frac{1}{4h^2} \sum_{i=1}^{L_x} \left[ f(\hat{x}_{k-1|k-1} + s_{k-1}^{x,i}, v_k, u_k) \right. \]

\[ - f(\hat{x}_{k-1|k-1} - s_{k-1}^{x,i}, v_k, u_k) \right] \]

\[ + \frac{h^2 - 1}{4h^4} \sum_{i=1}^{L_x} \left[ f(\hat{x}_{k-1|k-1} + s_{k-1}^{x,i}, v_k, u_k) \right. \]

\[ + f(\hat{x}_{k-1|k-1} - s_{k-1}^{x,i}, v_k, u_k) \right] \]

\[ + \frac{1}{4h^2} \sum_{i=1}^{L_v} \left[ f(\hat{x}_{k-1|k-1} + s_{k-1}^{v,i}, v_k + u_k) \right. \]

\[ - f(\hat{x}_{k-1|k-1}, v_k + s_{k-1}^{v,i}, u_k) \right] \]

\[ + \frac{h^2 - 1}{4h^4} \sum_{i=1}^{L_v} \left[ f(\hat{x}_{k-1|k-1} + s_{k-1}^{v,i}, v_k + u_k) \right. \]

\[ + f(\hat{x}_{k-1|k-1}, v_k + s_{k-1}^{v,i}, u_k) \right] \] (5.24)
5.2.4 Kolmogorov Smirnov Statistic Computation

The KS-test statistic \( D_n \) between univariate distribution \( F(x) \) and \( n \) i.i.d. observations with empirical cumulative distribution function \( F_n(x) \) is given by the following expression.

\[
D_n = \sup_{x \in \mathbb{R}} |F_n(x) - F(x)| \tag{5.25}
\]

where \( \sup \) is the supremum of the set of distances.

The particles generated at each iteration are ordered in a standard numbering scheme displayed in 5.3. Each particle set maintains a unique ordering through an iteration. The weights of each particle for each density considered are calculated using the expression

\[
w_{ki} = p(x_i^{\text{particle}} | x_{\text{current}}) \tag{5.26}
\]

where \( w_{ki} \) is the weight of the \( i^{th} \) particle at the \( k^{th} \) time step,

\( x_i^{\text{particle}} \) is the location of the \( i^{th} \) particle and

\( x_{\text{current}} \) is the estimate of the current distribution.

For distributions approximated by a single normal distribution, the weight of each particle is calculated as

\[
w_{ki} = \frac{1}{(2\pi)^{1.5}|P_{\text{current}}|} \exp\left(-\frac{1}{2}(x_{\text{current}} - x_i^{\text{particle}})^T P_{\text{current}}^{-1}(x_{\text{current}} - x_i^{\text{particle}})\right) \tag{5.27}
\]

where \( P_{\text{current}} \) is the covariance of the current distribution. For the GPF with \( N_{\text{GPF}} \) normal distributions,

\[
w_{ki} = \prod_{r=1}^{N_{\text{GPF}}} \frac{1}{(2\pi)^{1.5}|P_{r,\text{current}}|} \exp\left(-\frac{1}{2}(x_{\text{current}} - x_i^{\text{particle}})^T P_{r,\text{current}}^{-1}(x_{\text{current}} - x_i^{\text{particle}})\right) \tag{5.28}
\]

where each normal distribution of the Gaussian Particle Filter is represented by
A Comparison of SLAM Prediction Densities using the Kolmogorov Smirnov Statistic

\[ \mathcal{N}(x_{\text{current}}^r, P_{\text{current}}^r). \]

Each particle is equally weighted in the prior distribution with \( \frac{1}{N_{\text{part}}} \), where \( N_{\text{part}} \) is the number of particles.

The ordered set of weights associated with each probability distribution being analysed is considered as an observation, while the ordered set of weights associated with the particle filter is considered to be the ground truth for the current time step. Each ordered set of weights is compared with the ordered set of weights from the particle filter, to obtain the Kolmogorov-Smirnov statistic for each probability distribution for the current inputs. The inputs are then iterated, and the Kolmogorov-Smirnov statistic is calculated with the same procedure.

5.2.5 Pseudocode to Compute the Kolmogorov-Smirnov Statistic

**Algorithm 5** Computation of Kolmogorov Smirnov Statistic from prior density

```
for input \( u_k \) in set of inputs do
    generate ordered particles \( x_{p,k}^i \) in ordered particle set \( X_{p,k} \) of size \( N_{\text{part}} \) initialize particle weights \( w_{p,k}^i = \frac{1}{N_{\text{part}}} \) in ordered weighting set \( W_{p,k} \) for \( m^{th} \) algorithm to be tested in total set of \( M \) algorithms do
        calculate prior density \( p_m(x_{m,k}|u_k) \) for \( i^{th} \) particle in ordered set \( X_{p,k} \) do
            Calculate weight \( w_{m,k} = p(x_{p,k}^i|x_{m,k}) \) in ordered set of weights \( W_{m,k} \)
        end
        Calculate Kolmogorov-Smirnov Statistic \( KS_{m,k} \) between \( W_{m,k} \) and \( W_{p,k} \)
    end
end
```

Plot Kolmogorov-Smirnov Statistic arrays \( KS_m \) for all \( M \) algorithms for comparison
5.3 Simulation

The Ackermann model is used as the motion model with velocity \((v_k)\) and orientation change\((\delta \theta_k)\) as inputs. The inputs are incrementally increased and the KS-distances for the prediction densities generated from each algorithm are recorded.

The motion model for the Ackermann model is as follows:

\[
x_{k|k-1} = \begin{bmatrix} x_{k-1|k-1}(1) + v_k \cos(\delta \theta_k + x_{k-1|k-1}(3)) \\ x_{k-1|k-1}(2) + v_k \sin(\delta \theta_k + x_{k-1|k-1}(3)) \\ x_{k-1|k-1}(3) + \frac{v_k \delta t}{W_B} \sin(\delta \theta_k \delta t) \end{bmatrix}
\]

(5.29)

where \(W_B\) is the wheelbase length. A wheelbase of 0.8 m corresponding to the hardware setup shown in figure 5.1 is used as input.

The velocities and orientation changes are incremented simultaneously to obtain the new predictions. The velocity is changed from 0.1 m/sec to 2.5 m/sec with an increment of 0.45 m/sec, while the orientation change is increased from \(3\pi/150\) rad \((\approx 1.2)\) to \(3\pi/8\) rad \((\approx 67.5)\) with an increment of 0.1 rad.

The vehicle model covariances are \(\sigma_v^2 = 0.4 m^2/s^2\) and \(\sigma_{\delta \theta}^2 = 0.1 rad^2\).

The resultant trajectory is displayed in figure 5.2, along with the generated particles and the predicted estimate and covariance for each algorithm, as explained in the caption. The above procedure is carried out through 40 Monte Carlo runs, and the results are displayed in figure 5.5.

In order to measure the effect of platforms with different input variable noise statistics, Kolmogorov Smirnov statistic is plotted for velocity standard deviations from \(\sigma_v = 0.05 m/s \rightarrow 5.5 m/s\) and angle change standard deviation from \(0.04 rad \rightarrow 4.4 rad\). Additionally, the effect number of normal distributions in the GPF prediction density is analysed by plotting the Kolmogorov-Smirnov statistic for a \(N_{GPF}\) range of 3 \(\rightarrow 100\) at the same point.
in the trajectory. 40 Monte-Carlo runs are carried out for each simulation.

5.4 Simulation Result Analysis

5.4.1 Effect of Change in Inputs

In the initial portion (δ = 0 → 2.2 in the graph of figure 5.5 and figure 5.6), the EKF outperforms the UKF and the CDKF. Due to the limited nonlinearity of this portion, the particles cluster around the mean, as seen in figure 5.4. The EKF is inherently more linear than the UKF and the CDKF, the proof of which is beyond the scope of this chapter. Thus, in this initial portion, it behaves similar to the Kalman Filter, an optimal estimator for linear Gaussian noise, leading it to outperform the UKF and the CDKF.
5.4 Simulation Result Analysis

Fig. 5.5 Velocity and angle inputs are increased simultaneously and the Kolmogorov-Smirnov statistic is calculated for each iteration. The x-axis displays a multiplicative factor for a base velocity of 0.05 m/sec and a base angle change of 0.1 rad. The vehicle model covariances are $\sigma_v^2 = 0.4 m^2/ s^2$ and $\sigma_{\delta\theta}^2 = 0.1 rad^2/ s^2$.

Fig. 5.6 Only Velocity input is increased and the Kolmogorov-Smirnov statistic is calculated for each iteration. The x-axis displays a multiplicative factor for a base velocity of 0.05 m/sec. The vehicle model covariances are $\sigma_v^2 = 0.4 m^2/ s^2$ and $\sigma_{\delta\theta}^2 = 0.1 rad^2/ s^2$. 
Fig. 5.7 Only angle input is increased and the Kolmogorov-Smirnov statistic is calculated for each iteration. The x-axis displays a multiplicative factor for a base angle change of 0.1 rad. The vehicle model covariances are $\sigma_v^2 = 0.4 m^2/s^2$ and $\sigma_{\delta\theta}^2 = 0.1 rad^2/s^2$.

In the later portion of the simulation ($\delta = 2.2 \rightarrow 10$ in figures 5.5 and 5.6), the increased nonlinearity in the vehicle noise leads to the UKF and the CDKF outperforming the EKF again.

In both portions, the GPF prediction density is the nearest to the ground truth represented by the particles. In the earlier portion, the normal distributions in the GPF prediction density have significant overlap with each other, leading the filter to behave as an EKF. In the later portion, however, the normal distributions spread out similar to the particle filter, leading to a higher portion of the particles to be covered by the GPF prediction density.

In the case of only the angle input changing, with constant velocity input, the EKF performs poorly for initial values ($\delta = 0 \rightarrow 1.8$ in figure 5.7), in comparison to the UKF and CDKF algorithms. For later values ($\delta = 1.8 \rightarrow 10$ in figure 5.7), the EKF clearly outperforms both the UKF and CDKF algorithms.
5.4 Simulation Result Analysis

Fig. 5.8 Change in Kolmogorov-Smirnov Statistic when velocity and angle standard deviation are simultaneously changed. The x-axis displays a multiplicative factor for a base velocity standard deviation of 0.5m/s and a base angle change standard deviation of 0.4rad. $N_{GPF} = 50$

The effect of change in angle inputs on the GPF algorithm is negligible, and almost the same Kolmogorov-Smirnov statistic value is maintained throughout the set of inputs iterated against [4].

5.4.2 Effect of Change in Standard Deviations

For small standard deviations, the algorithms in increasing order of performance are the EKF, CDKF and the UKF as seen in figures 5.8, 5.9 and 5.10. As the standard deviations are increased, the EKF begins outperforming the UKF and the CDKF for all sets of standard deviation changes. All three algorithms display similar increases in the Kolmogorov-Smirnov statistic, however, with the rate of increase being the least in the case of the EKF, and the maximum in the case of the UKF algorithm.

Conversely, the GPF prediction density performs at par, or worse than the other three
Fig. 5.9 Change in Kolmogorov-Smirnov Statistic when only velocity standard deviation is changed. The x-axis displays a multiplicative factor for a base velocity standard deviation of $0.5 m/s$. $N_{GPF} = 50$

Fig. 5.10 Change in Kolmogorov-Smirnov Statistic when only angle change standard deviation is changed. The x-axis displays a multiplicative factor for a base angle change standard deviation of $0.4 rad$. $N_{GPF} = 50$
algorithms for low input variable standard deviations, but its performance improves as the standard deviations are increased, till a stable value of the KS-statistic is reached and maintained. The rate of decrease of the KS-statistic in the case of the GPF algorithm can be observed to have inverse-exponential characteristics in all three cases of input variable standard deviation changes. A possible reason for this is the similarity of the GPF prediction density generation method with the particle filter prediction density generation method.

5.4.3 Effect of Change in Number of Gaussians for the Gaussian Particle Filter

As expected, the KS-statistic of the EKF, UKF and CDKF algorithms are unaffected by the change in number of Gaussians in the predicted density, since their prediction densities are approximated as a single normal distribution. On the other hand, the performance of the GPF improves proportionally to the number of Gaussian distributions used, till the $N_{GPF} = 50$ mark is reached, subsequent to which, the performance achieves an upper bound and remains stable. The performance of the GPF displays a clear inverse exponential characteristic, as seen in figure 5.11.

5.5 Experimental Results

The robotic platform is navigated through three trajectories, as shown in figures 5.12, 5.14 and 5.16 and observations from the visual odometry and the LiDAR sensor are recorded. This navigation is performed 20 times to ensure the effect of random noise is cancelled out, while the overall noise statistics of the sensors are maintained.

The velocity and angle change inputs are obtained from the visual odometry. The LiDAR-based pose estimate obtained from [53] is used as the ground truth. The experiment
A Comparison of SLAM Prediction Densities using the Kolmogorov Smirnov Statistic

Fig. 5.11 Change in Kolmogorov-Smirnov Statistic when number of Gaussians for the Gaussian particle filter prediction density is changed. The x-axis displays a multiplicative factor for a base $N_{GPF} = 3$. The vehicle model covariances are $\sigma_v^2 = 0.4m^2/s^2$ and $\sigma_{\theta}^2 = 0.1 rad^2$.

Fig. 5.12 Trajectory of robot in experiment. The figure shows a map obtained from the ROS Hector mapping package. The blue arrows depict the trajectory predicted by the visual odometry, while the red arrow depicts the current ground truth, obtained from the Hector mapping estimate. The background grid has a unit size of $1m$. 
5.5 Experimental Results

Fig. 5.13 Prediction density likelihood obtained from experiment. The x-axes displays a multiplicative factor with a base velocity of 0.1m/s and angle change of $\pi/50$

is performed in an indoor environment, which is maintained as uncomplicated and static as possible, in order to guarantee maximum accuracy of the LiDAR-based pose estimate([3]).

The likelihood of the prediction densities of each algorithm ($p(\hat{x}_{k|k-1}|x_k)$) is calculated using the ground truth. Since, in the real world, the true prediction density of the vehicle cannot be obtained to calculate the KS-statistic, this expression is used instead as a basis of comparison.

In the experiment depicted in 5.12, translational change (corresponding to velocity change) from $0 \rightarrow 2.5m$ and an angle change from $0 \rightarrow \pi/2$. The result is plotted in figure 5.13.

The experiment depicted in 5.14 has translational change corresponding to change in velocity input from $0 \rightarrow 2.5m$, and no change in angle inputs. Results obtained are plotted in 5.15.

The input parameters for the experiment shown in 5.16 has translational change corresponding to change in velocity input from $0 \rightarrow 1.0m$, and an angle change from $0 \rightarrow \pi/2$. The result is plotted in figure 5.17.
Fig. 5.14 Trajectory of robot in experiment with no angle change. The figure shows a map obtained from the ROS Hector mapping package. The blue arrows depict the trajectory predicted by the visual odometry, while the red arrow depicts the current ground truth, obtained from the Hector mapping estimate. The background grid has a unit size of 1m.

Fig. 5.15 Prediction density likelihood obtained from experiment with no angle change. The x-axes displays a multiplicative factor with a base velocity of 0.1m/s with no angle change.
The platform noise covariance values of $\sigma_v^2 = 0.4m^2/s^2$ and $\sigma_{\delta\theta}^2 = 0.1rad^2$ are used.

The comparison is performed between the GPF, EKF, UKF and CDKF algorithms. The particle filter prediction density consists of a summation of discrete point particles. Since the expression $p(\hat{x}_{k|k-1}|x_k)$ cannot be calculated for such a prediction density, the particle filter is left out of the comparison.

As observed in figure 5.13, the prediction densities of all the algorithms are negatively affected by when inputs are increased. This observation is as expected, since convergence of state estimation algorithms are contingent on frequent observation inputs. The Gaussian Particle Filter outperforms the Extended Kalman Filter, the Unscented Kalman Filter and the Central Difference Kalman Filter by a large margin initially. However, the drop in performance is similarly higher for the GPF, as compared to the UKF, EK and CDKF algorithms, until a similar lower bound for all three algorithms is obtained asymptotically.
Fig. 5.17 Prediction density likelihood obtained from experiment with small angle change. The x-axes displays a multiplicative factor with a base velocity of $0.04 \text{m/s}$ and angle change of $\pi/50$.

### 5.6 Conclusion

This chapter demonstrates a method to compare the prediction densities of the Extended Kalman Filter, Unscented Kalman Filter, Central Difference Kalman Filter and the Gaussian Particle Filter algorithms used for robot localization using the Kolmogorov-Smirnov test statistic.

The EKF outperforms the UKF and the CDKF in the initial portions of the test with more linear change in noise, while the UKF and CDKF algorithms outperform the EKF in later portions of the test with higher nonlinearity in the trajectory.

Change in noise statistics of the vehicle demonstrate an inverse relation in the performance of the algorithms, since the EKF density is able to encapsulate a higher proportion of particles generated, with higher input noise covariances.

The GPF outperforms all three algorithms, and is the closest to the particle filter ground truth. This is possibly because the particle filter and the GPF share similar prediction density generation schema. This analysis is strengthened by the observation that the GPF demonstrates an inverse exponential curve in performance over all the parameter measured.
The prediction densities were analysed over a range of noise statistics using the data obtained from a specific robot model. Further analysis employing a larger variety of robotic platforms would be greatly beneficial to the comparison schema.
Chapter 6

A Gaussian Particle Filter based Factorised Solution to the Simultaneous Localization And Mapping problem

6.1 Introduction

One of the biggest drawbacks of the particle filter, is the problem of degeneracy of the particles which occurs when the particles diverge over a long period of time, leading to poor estimates. The resampling sub-algorithm is the solution to this problem. However, sustained resampling causes massive sample impoverishment resulting in a few particles obtaining most of the weight. Due to this, lower weighted competing hypotheses get filtered out, allowing random clutter in observations to cause divergence in estimates [65].

Owing to its dependence on the particle filter, the FastSLAM inherits both these drawbacks [40]. Furthermore, lack of correlation between landmarks decreases convergence rate of the map estimate, unlike the Extended Kalman Filter based SLAM algorithm.

A recently formulated solution to these problems is the Gaussian Particle Filtering al-
algorithm which has been successfully implemented in the target tracking community [32], which is based on the fact that, given the mean and the covariance of any probability distribution, the normal distribution maximizes entropy of the random variable, and is the least informative distribution. The implication of this property is that mean and covariance of any probability density can be tracked using a Gaussian probability distribution function. It has been proven that any probability density can be accurately represented by an infinite weighted Sum of Gaussians [44], which is analogous to the primary assumption made while formulating the particle filter. The Gaussian Particle Filter [46] and the Gaussian Sum Particle Filter [13] utilize this assumption by representing the prior in each iteration with a weighted sum of Gaussian densities. Each particle is then weighted according to the observation received, and updated according to the Extended Kalman Filter update algorithms. The posterior estimate is the weighted sum of all the particle posteriors. This avoids the problem of particle impoverishment.

The Gaussian Particle Filter also offers the following advantages, as derived in [46], which make it well suited as the basis of a SLAM formulation:

1. Unlike the EKF, the assumption of additive Gaussian noise can be discarded for the GPF. Thus nonlinear motion models using non-additive non-Gaussian noises can also be implemented using the GPF.

2. The GPF tracks the mean and covariance of the state, which result it in being a very computationally efficient algorithm, as shall be displayed further in this chapter.

3. The GPF approximates the posterior distribution as Gaussians, due to which it is possible to discard resampling. This results in information being conserved in the process, which would have otherwise be lost by components as they were resampled. As noted in [40], discarding the resampling process also improves consistency of the vehicle pose in the FastSLAM algorithm as well.
4. Modifying the importance sampling procedure for the GPF, as mentioned in [46], makes it possible to propagate a higher moment than the covariance as well.

In the next section, we will describe the formulation of the Gaussian Particle Filter based FastSLAM algorithm. In Section 6.3, we will simulate the GPF-FastSLAM algorithm, and compare the estimate results and computational efficiency with the EKF-SLAM, FastSLAM and UKF-SLAM algorithms. Section 6.4 will implement the GPF-FastSLAM algorithm using high clutter-affected data obtained from a marine environment, and compare the estimates obtained with those obtained via the FastSLAM and EKF-SLAM algorithms.

Section 6.5 will analyse the results obtained, and the subsequent section will detail the conclusions reached, as detailed in [62].

### 6.2 Formulation

The FastSLAM algorithm, as originally formulated, represents the vehicle pose estimate as a set of particles, each of which maintain an independent map consisting of a set of independent Extended Kalman Filters. The particles are sampled from the probability density according to the following equation:

$$
\mathbf{x}_k^m = p(\mathbf{x}_k | u_k, \mathbf{x}_{k-1}^m) 
$$

(6.1)

where:

- \( \mathbf{x}_k \) and \( \mathbf{x}_{k-1} \) are the vehicle pose estimates at time steps \( k \) and \( k-1 \) respectively,
- \( m \) represents particle number \( m \) out of a total of \( N \) particles and
- \( u_k \) is the control input provided to the vehicle at time step \( k \)

Using the transition prior as the Importance Sampling function, the weight of each par-
A Gaussian Particle Filter based Factorised Solution to the Simultaneous Localization And Mapping problem

das after the next set of observations is given by

$$w_k[m] = p(z_k | x_k[m])$$  \hspace{1cm} (6.2)

where

$z_k$ is the set of observations at time $k$ and
$w_k[m]$ is the weight of particle number $m$ at the current time step.

In feature based FastSLAM, this expression translates to the following equation:

$$w_k[m] = w_{k-1}[m] \prod_{i=1}^{K} p(\theta_i | x_k[m])$$  \hspace{1cm} (6.3)

where

$K$ is the number of features in the map of particle number $m$
$\theta_i$ is the location of feature number $i$

The particles are then resampled according to these weights, and the maps of the surviving particles are updated by the Extended Kalman filter update equations.

Unlike the Extended Kalman Filter based SLAM, FastSLAM assumes the feature estimate of each particle is uncorrelated with the other feature estimates. Hence, each feature would have to be observed before the covariance matrix of its estimate is updated. This is a serious drawback in large maps where the vehicle is unable to traverse multiple times.

The FastSLAM algorithm, being based on the particle filter, also inherits the drawbacks of the particle filter. Over-resampling of particles results in particle impoverishment, causing a few particles to inherit most of the weight, and resulting in viable particles being discarded due to clutter observations. Multiple simultaneous iterations affected by this problem will cause a divergence in the estimate of the vehicle.
In order to address these drawbacks with the FastSLAM algorithm, we propose a solution based on the Gaussian Particle Filter. Any probability distribution function can be expressed with sufficient accuracy by a sufficiently large number of normal distributions. The Gaussian Particle Filter algorithm utilizes this observation by propagating the prior distribution of the state estimate with a weighted sum of Gaussian distributions, which are then updated with observations using Extended Kalman Filter update equations. The weighted Gaussian hypotheses are then summed up to obtain the final estimate posterior.

This prior distribution is then propagated as a mixture of $G$ Gaussians according to the following equation:

$$x_{k|k-1} = \sum_{i=1}^{G} \frac{1}{G} x_{k|k-1}^{[i]} \quad (6.4)$$

where

$$x_{k|k-1}^{[i]} \sim \mathcal{N}(x_k; u_k, x_{k-1|k-1}) \quad (6.5)$$

and $G$ is the number of constituent Gaussian distributions. The number of Gaussians depends on the dynamics of the motion model. The greater the nonlinearity, the higher the number of Gaussians required.

On receiving the observations $z_k$, we calculate the weight of each Gaussian sample using the following expression:

$$w_{k|k}^{[i]} = \frac{p(z_k|x_{k|k-1}^{[i]}, \mathcal{N}(x_{k|k-1}^{[i]}, x_{k-1|k-1}, P_{k-1|k-1}))}{\pi(x_{k|k-1}^{[i]}|z_0:k)} \quad (6.6)$$
A Gaussian Particle Filter based Factorised Solution to the Simultaneous Localization And Mapping problem

where

$P_{k-1|k-1}$ is the prior state covariance,

$\pi(x_k^{[i]}|z_{0:k})$ is the importance sampling density function

In our implementation, we use $p(x_k|z_{0:k-1})$, as our importance sampling function $\pi(x_k^{[i]}|z_{0:k})$, as suggested in [46].

The state estimate and covariance are updated using the following equations:

$$x_{k|k} = \sum_{j=1}^{G} w_k^{[j]} x_k^{[j]} \quad (6.7)$$

$$P_{k|k} = \sum_{j=1}^{G} w_k^{[j]} (x_{k|k-1}^{[j]}(x_k^{[j]})(x_k^{[j]}))^T \quad (6.8)$$

6.3 Implementation

In our implementation of the GPF-based FastSLAM algorithm, we propagate $G$ normal distributions by simulating $G$ noise values, $\delta \theta_{ki}$ and $V_{ki}$

$$[\delta \theta_k^{[i]} \ V_k^{[i]}]^T \sim \mathcal{N}(0, Q_k) \quad (6.9)$$

which are then used in the standard wheelbase vehicle dynamics model (as shown in Fig.3.1) as follows:

$$x_{k|k-1}^{[i]}(1) = x_{k-1}(1) + V_k^{[i]} dt \cos(x_{k-1}(3) + \delta \theta_k^{[i]}) \quad (6.10)$$

$$x_{k|k-1}^{[i]}(2) = x_{k-1}(2) + V_k^{[i]} dt \sin(x_{k-1}(3) + \delta \theta_k^{[i]}) \quad (6.11)$$
Fig. 6.1 Simulated Trajectory. The Trajectory taken by the vehicle shown in black, the vehicle is represented as a green triangle. The real features are displayed as blue crosses. The current scan associations are shown as red lines.

\[
x^{[i]}_{(k|k-1)}(3) = x_{k-1}(3) + \frac{V_k^{[i]} dt}{W_B} \sin(\delta\theta_k^{[i]})
\]  

\[ (6.12) \]

where \( i \) depicts the \( i^{th} \) normal distribution, out of a total of \( G \) distributions. The \( G \) Gaussians are initially equally weighted as:

\[
w^{(k|k-1)i} = \frac{1}{G}
\]

\[ (6.13) \]

The area of the prediction density with higher probability contains a greater concentration of Gaussians, as compared to outlying areas with lesser probability. Thus, the weighted sum of the equally weighted Gaussians combined with their respective sampled locations equate to the required predictions. In our simulations, we assume the standard deviations for the
velocity and orientation change as 5 m/sec, and 20 degrees respectively. We implement the GPF-FastSLAM algorithm in the trajectory depicted in Fig.6.1. We compare the computational time, trajectory and mapping errors of the GPF to those caused by EKF-SLAM, FastSLAM and UKF-SLAM using similar configurations. The vehicle was driven through a predetermined trajectory through an environment containing simulated features. The estimation was carried out using the FastSLAM, EKF-SLAM and GPF-FastSLAM algorithms, and the estimation errors were compared. This operation was performed 50 times, and the resulting errors were averaged out.

As seen in Fig.6.2, the GPF-FastSLAM algorithm outperforms the EKF-SLAM and the FastSLAM algorithms while performing state estimations, for the same parameters. The EKF-SLAM algorithm performance improves drastically towards the end of the loop, when a large number of previously-observed features help the estimator converge. However, in more realistic scenarios dealing with presence of clutter and false detections which can cause the EKF-SLAM state estimate to diverge, algorithms such as FastSLAM and GPF-
6.3 Implementation

Fig. 6.3 Comparison of Computational Time after 50 Simulated Runs

Algorithm 6 GPF-SLAM algorithm

Initialization \((x_0, P_0, Q_0, N_{\exp})\)

for all \(x_k\) do

for Gaussians \(G = 1 \text{to} N_{\text{GPF}}\) do

\[ v_{k,G} = [\delta \theta_k^G, V_k^G]^T \sim \mathcal{N}(0, Q_k) \]

predict

\[ x_{k|k-1,G} = f(x_{k-1}, u_k, v_{k,G}) \]

\[ F_{k,G} = \frac{\partial}{\partial x} f(x, u)|_{x_{k|k-1,G}} \]

\[ M_{k,G} = \frac{\partial}{\partial v} f(x, u)|_{x_{k|k-1,G}} \]

\[ P_{k|k-1,G} = F_{k,G} P_{k-1,k-1} F_{k,G}^T + M_{k,G} Q_k M_{k,G}^T \]

Weighting \(w_{k|k-1,G} = \frac{1}{N_{\text{GPF}}} \)

predicted observations \(\hat{z}_{k,G} = h_{k,G}(x_{k|k-1,G}, s)\)

obtain real observations \(z_k\)

update

Calculate weight \(w_{k,G} = w_{k|k-1,G} \frac{p(z_k|x_{k|k-1,G})}{p(z_k|x_{k|k-1,G}^E)} \)

Update estimates \(x_k|k = \frac{\sum_{G=1}^{N_{\text{GPF}}} w_{k,G} x_{k|k-1,G}}{\sum_{G=1}^{N_{\text{GPF}}} w_{k,G}} \)

Update covariance \(P_k|k = \frac{\sum_{G=1}^{N_{\text{GPF}}} w_{k,G}^2 x_{k|k-1,G}^2 + P_{k|k-1,G}}{\sum_{G=1}^{N_{\text{GPF}}} w_{k,G}^2} \)

end

end
A Gaussian Particle Filter based Factorised Solution to the Simultaneous Localization And Mapping problem

FastSLAM, with multiple state hypotheses, can perform much better.

The map estimation of the GPF-FastSLAM algorithm is also much better than either the EKF-SLAM or the FastSLAM algorithm.

This is due to the availability of multiple map hypotheses, similar to the FastSLAM algorithm, and the correlation of the landmarks in the map leading to better convergence, similar to the EKF-SLAM algorithm.

The computational time is displayed in Fig.6. As expected, while it does not perform as well as the EKF-SLAM algorithm, it is much faster than the FastSLAM algorithm for similar parameters. Due to the presence of a fewer number of features in the environment, such as the test scenario, the augmentation of the EKF-SLAM state estimate is minimal, resulting in lower computational time for the state estimate.

6.4 Experimental Results

The GPF-SLAM algorithm was also benchmarked against the EKF-SLAM and FastSLAM algorithm using the marine dataset described in chapters 4.2.3 and 4.3.2. Fig.6.4 displays the comparison of localization errors obtained from the GPF-FastSLAM, EKF-SLAM and the FastSLAM algorithms. Since the GPF-FastSLAM algorithm performs updates by correlating the landmark estimates together, similar to the EKF-SLAM algorithm, we observe that the errors of both algorithms are similar, with the GPF-FastSLAM algorithm outperforming the EKF-SLAM algorithm due to the usage of multiple hypotheses, inherent in the formulation.

The FastSLAM algorithm performs similar to the EKF-SLAM and GPF-FastSLAM algorithm initially, but as the mis-detections and false features build up due to the presence of clutter, the localization error accumulates, causing the estimate to begin diverging. Fig.6.5 shows the comparison of time taken for all 3 algorithms, and we see here that GPF-
6.4 Experimental Results

Fig. 6.4 Localization Error Comparison

Fig. 6.5 Computational Time Comparison using Data from Kayak
FastSLAM is much faster than the FastSLAM algorithm, and comparable in speed to the EKF-SLAM algorithm.

6.5 Conclusion

The Gaussian Particle Filter-FastSLAM algorithm combines the best features of both the EKF-SLAM algorithm and the FastSLAM algorithm, leading to better convergence of both the vehicle state estimate and the map estimate. It is also comparable in speed to the EKF-SLAM algorithm, and much faster than the FastSLAM algorithm. It outperforms both algorithms in environments with high clutter, and large nonlinearities in motion models, as displayed by its performance in the marine environment.
Chapter 7

A GPF based Adaptive SLAM Algorithm

7.1 Introduction

As seen in the previous chapters, the Gaussian Particle Filter (GPF) algorithm estimates the SLAM posterior more accurately than other comparable state of the art SLAM algorithms due to its approximation of the motion model prior as a sum of normal distributions. The speed and flexibility offered by this formulation allows the algorithm to quickly and efficiently approximate a wide spectrum of motion model noise, making it a very promising candidate as the basis of formulation for an adaptive sensor fusion algorithm for Simultaneous Localization and Mapping.

To this end, the Factorized Solution to the Simultaneous Localization and Mapping (FastSLAM) algorithm is reformulated with an adaptive covariance estimation algorithm based on the GPF, to obtain the Adaptive Gaussian Particle Filter-based Factorised Solution to Simultaneous Localization and Mapping (AGPF-FastSLAM) algorithm([2]).

The AGPF-FastSLAM algorithm is formulated in the next section, with a pseudocode to describe the algorithm. A simulation is described in the following section, and the results obtained are analyzed. The subsequent section will describe the hardware setup deployed to
obtain experimental data from a marine environment, as well as provide a detailed analysis of the results obtained. The final section will present conclusions to the chapter.

7.2 Formulation

The process model noise statistics is divided into \( N_G \) normal distributions \( \sim \mathcal{N}(\mu_{l,k}^*, Q_{l,k}) \) generated using the following expression:

\[
\mu_{l,k} \sim \mathcal{N}(\hat{\mu}_{k-1}, Q_{k-1}); \quad (l = 1 \rightarrow G)
\]

\[
Q_{l,k}^* = \frac{w^2 g(l,k-1) \times Q_{k-1}}{\sum_{l=1}^{G} w^2 g(l,k-1)}
\]

The following relation is maintained:

\[
\sum_{l=1}^{G} \mathcal{N}(\mu_{l,k}, Q_{l,k}^*) = \mathcal{N}(0, Q_{k-1})
\]

The asterisk (*) in the nomenclature is used to specify the noise statistics calculated for the current time step \( k \).

Each of the \( G \) normal distributions generated is considered an independent hypothesis of the process model noise statistics, and is used to generate particles. A bank of \( G \) parallel independent FastSLAM estimation algorithms are maintained using the generated model statistics.

The \( N_{Particles} \) particles are equally divided between the \( G \) FastSLAM estimators, resulting in each estimator employing \( N_{Particles}/G \) independent particle hypotheses. The weighted average of the innovations \( v_k \) of all the particles in each estimator is considered to be the
'observed measurement mean' of the estimator for the current time step.

\[ \nu_{k,i} = \frac{\sum_{j=1}^{N_{\text{particles}}} w_{p(j,i)} \nu_{k,j,i}}{\sum_{j=1}^{N_{\text{particles}}} w_{p(j,i)}} \]  

(7.4)

Where \( i \) is the index of the noise statistic hypothesis, \( w_{p(k,j,i)} \) is the weight of the \( j^h \) particle, and \( \nu_{k,j,i} \) is its innovation.

The noise statistics mean from the estimator is calculated from the inverse measurement model \( h^{-1}() \), given by the following equation:

\[ \mu_{k,i} = h^{-1}(\nu_{k,i}) \]  

(7.5)

The weight of the gaussian distribution hypothesis is calculated according to the following equation:

\[ w_{g(k,i)} = \frac{1}{\sqrt{2\pi|Q_{i,k-1}|}} \exp\left(-\frac{1}{2} \mu_{k,i} \times Q_{i,k-1}^{-1} \mu_{k,i}^T\right) \]  

(7.6)

The final state estimate is the weighted sum of the estimates of all the covariance hypotheses calculated by the following expression.

\[ \hat{x}_k = \frac{\sum_{i=1}^{G} w_{g(k,i)} \hat{x}_{k,i}}{\sum_{j=1}^{N_{\text{particles}}} w_{g(k,i)}} \]  

(7.7)

The model bias is estimated by taking the weighted average of all the covariance hypotheses as follows:

\[ \hat{\mu}_k = \frac{\sum_{i=1}^{G} w_{g(k,i)} \times \mu_{i,k-1}}{\sum_{i=1}^{G} w_{g(k,i)}} \]  

(7.8)

The model covariance is estimated by taking the weighted average of all the covariance hypotheses as follows:

\[ \hat{Q}_k = \frac{\sum_{i=1}^{G} w^2_{g(k,i)} \times Q_{i,k-1}}{\sum_{i=1}^{G} w^2_{g(k,i)}} \]  

(7.9)
The estimated covariance will be fed into the algorithm as the process model covariance in the next step.

The estimated bias and covariance will be fed into the algorithm as the process model bias and covariance in the next step.

The algorithm can be summarized with the pseudocode presented below.

```plaintext
while r do
  o
end
bot path is not completed
for i = 1 G do
  \( \mu_{i,k} \sim \mathcal{N}(\hat{\mu}_{k-1}, \hat{Q}_{k-1}) \)
  \( Q_{i,k-1} \leftarrow \frac{1}{\sqrt{G}} \hat{Q}_{k-1} \)
  for j = 1 \( \frac{N_{\text{Particles}}}{G} \) do
    \( x_{k|k-1,j,i} \leftarrow f(x_{k-1|k-1,j,i}, u_k, \mu_{i,k}) \)
    \( \tilde{z}_{k,j,i} \leftarrow h(x_{k|k-1,j,i}, w_{k,j}) \)
    \( v_{k,j,i} = z_k - \tilde{z}_{k,j,i} \)
    \( H_{k,j,i} \leftarrow \frac{\partial}{\partial x} h(x,w) \big|_{x=x_{k|k-1,j,i}} \)
    \( S_{k,j,i} \leftarrow H \times R \times H^T \)
    \( w_{p(k,j,i)} \leftarrow \frac{1}{\sqrt{2\pi|S_{k,j,i}|}} \exp \left( \frac{1}{2} v_{k,j,i} \times S_{k,j,i}^{-1} v_{k,j,i} \right) \)
  end
RESAMPLE Particles of current noise statistic set
  \( \hat{x}_{k,i} = \frac{\sum_{j=1}^{N_{\text{Particles}}} w_{p(k,j,i)} x_{k|k-1,j,i}}{\sum_{j=1}^{N_{\text{Particles}}} w_{p(j,i)}} \)
  \( v_{k,i} = \frac{\sum_{j=1}^{N_{\text{Particles}}} w_{p(j,i)} v_{k,j,i}}{\sum_{j=1}^{N_{\text{Particles}}} w_{p(j,i)}} \)
  \( \mu_{k,i} = h^{-1}(v_{k,i}) \)
  \( w_{g(k,i)} \leftarrow \frac{1}{\sqrt{2\pi|Q_{i,k-1}|}} \exp \left( \frac{1}{2} \mu_{k,i} \times Q_{i,k-1}^{-1} \mu_{k,i} \right) \)
end
\( \hat{\mu}_k = \frac{\sum_{i=1}^{G} w_{g(k,i)} \times \mu_{i,k-1}}{\sum_{i=1}^{G} w_{g(k,i)}} \)
\( \hat{Q}_k = \frac{\sum_{i=1}^{G} w_{g(k,i)}^2 \times Q_{i,k-1}}{\sum_{i=1}^{G} w_{g(k,i)}^2} \)
\( \hat{x}_k = \frac{\sum_{i=1}^{G} w_{g(k,i)} \hat{x}_{k,i}}{\sum_{i=1}^{N_{\text{Particles}}} w_{g(k,i)}} \)
```
7.3 Simulation

A simulated run was carried out with an unmanned ground vehicle in a preset trajectory.

The trajectory shown in fig. 7.1 was used. Five simulations were carried out as follows:

1. with the model bias $b_k = 0.2 * u_k$, the simulated control input

2. with the model bias exhibits gaussian statistics $b_k \sim \mathcal{N}(0.2b_k, 0.15Q)$, where $Q$ is the simulated process covariance

3. with the model covariance $\sigma_{model} = 1.3 * \sigma_{sim}$, the simulated process covariance

4. with the model covariance $\sigma_{model} = \frac{1}{1.3} \sigma_{sim}$, the simulated process covariance

Fig. 7.1 Simulated Trajectory, with the axes marked in metres. The green asterisks are the feature location ground truths, the red dots adjacent to the asterisks are the feature location estimates. The red triangle is the vehicle pose ground truth, while the adjacent green triangle is the pose estimate. Additive bias is added to the noise in the part of the trajectory shown in green colour, while subtractive bias is added to the noise in the part of the trajectory shown in red colour.
Fig. 7.2 Initial distribution of estimated map with newly initialized process model gaussian hypotheses

Fig. 7.3 Converged distributions of estimated map with converging process model gaussian hypotheses
5. a control simulation was also carried out in which the model covariance $\sigma_{\text{model}} = \sigma_{\text{sim}}$, the simulated process covariance and the model bias $b_k = 0$ (that is, the process model noise statistics were accurately input into the algorithm)

The Ackermann model is used as the motion model with velocity ($v_k$) and orientation change ($\delta \theta_k$) as inputs for a time step of length $\delta t$. The motion model for the Ackermann model is as follows:

$$
x_{k|k-1} = 
\begin{bmatrix}
  x_{k-1|k-1}(1) + v_k \cos(\delta \theta_k + x_{k-1|k-1}(3)) \\
  x_{k-1|k-1}(2) + v_k \sin(\delta \theta_k + x_{k-1|k-1}(3)) \\
  x_{k-1|k-1}(3) + \frac{v_k}{W_B} \sin(\delta \theta_k \delta t)
\end{bmatrix}
$$

where $W_B$ is the wheelbase length, in this case 1.2 m corresponding to the hardware setup. Each simulated trajectory run was carried out for 50 Monte Carlo runs, to ensure outlier results were averaged out. The AEKF-SLAM, AGPF-FastSLAM, and FastSLAM algorithms were compared compared in five configurations:

- AEKF-SLAM with small window size 5 inputs
- AEKF-SLAM with medium window size 25 inputs
- AEKF-SLAM with large window size 45 inputs
- AGPF-FastSLAM algorithm with the same number of particles as the other algorithms, divided among 5 normal distributions
- FastSLAM algorithm

The RMS Error for the SLAM pose was used as a measure of the algorithm. The ex-
expression to calculate the RMS error is as follows:

\[
\tilde{x}_{k|k-1} = \begin{bmatrix}
    x_k^2(1) - \hat{x}_{k|k}(1) \\
    x_k^2(2) - \hat{x}_{k|k}(2) \\
    x_k^2(3) - \hat{x}_{k|k}(3)
\end{bmatrix}
\]  

(7.11)

As can be seen in figs. 7.4, 7.5, 7.6, 7.7, 7.8, the trends for all five types of simulations are the same. A few observations on the results are as follows:

- The FastSLAM algorithm outperforms all configurations of AEKF-SLAM algorithms when the process model is perfectly simulated.

- For all cases of mismodeled process model noise statistics, the performance of the FastSLAM algorithms lies between the performance of the AEKF-SLAM algorithms with small and medium window sizes.

- The performance of the AEKF-SLAM algorithm with large window size is the poorest of all five algorithms being compared. The algorithm performs better initially when there are small number of corrections of the process model covariance. However,
Fig. 7.5 Estimation with fixed drift

Fig. 7.6 Estimation with random drift
Fig. 7.7 Estimation with model covariance larger than actual covariance

Fig. 7.8 Estimation with model covariance smaller than actual covariance
when the number of corrections increases, the slow frequency of feedback results in the algorithm overcorrecting the covariance using information from the previous window

- The AGPF-FastSLAM outperforms all the other algorithms, including the FastSLAM algorithm, in all three scenarios.

- The AGPF algorithm clearly displays convergence of the model statistics. In fig. 7.2, multiple hypotheses for each landmark can be seen at the start of the simulation run, depicting the independent maps of the multiple hypotheses. Towards the end of the simulation run, as seen in fig. 7.3, the maps are clearly overlapping, signifying the convergence of the initial multiple motion model hypotheses.

### 7.4 Experimental Results

After being favorably compared against similar state-of-the-art algorithms in a simulated ground vehicle, the AGPF algorithm was benchmarked in an environment of greater complexity to better evaluate the efficacy of the algorithm, using the marine dataset described in chapter 4.2.3.

The position estimates obtained were compared against GPS ground truth. Five normal distributions were used as process model covariance hypotheses for the AGPF-FastSLAM algorithm. To maintain a uniform basis of comparison, an equal number of particles were used for the FastSLAM, AEKF-FastSLAM and the AGPF-FastSLAM algorithms, with the exception that the particles were uniformly divided among the normal distributions for covariance estimation in the AGPF-FastSLAM algorithm.

The comparisons between the Root Mean Square Errors are shown in fig. 7.9. In the EKF-SLAM, errors in pose estimation due to environmental biases are compounded due to
correlation between feature estimates, resulting in divergence of the SLAM estimates due to bad feature associations, causing the EKF-SLAM algorithm to perform the poorest of all four algorithms compared. In comparison, the FastSLAM algorithm, though suffering from the same errors in feature associations and pose estimate divergence, outperforms the EKF-SLAM algorithm due to the lack of correlation between the feature estimates.

The AEKF-FastSLAM outperforms both the static SLAM algorithms compared (EKF-SLAM and FastSLAM). Since a low window size of 5 steps was used, the frequent correction of the process model covariance matrix results in the pose estimates closely matching the estimates of the FastSLAM algorithm.

The AGPF-FastSLAM algorithm displays a similar trend in estimates as the FastSLAM and AEKF-FastSLAM algorithm, since it is structured as bank of FastSLAM algorithms performing SLAM estimates in parallel. However, the AGPF-FastSLAM algorithm clearly
outperforms both the other FastSLAM based algorithms with the same number of particles.

7.5 Conclusion

The AGPF-FastSLAM algorithm with online covariance estimation outperforms all SLAM algorithms benchmarked against, composed of both - static as well as adaptive sensor fusion algorithms.

The AGPF-FastSLAM algorithm inherits the FastSLAM map representation, allowing it to efficiently scale up the number of features, as well as decorrelate the errors arising from misestimation and mis-association of individual features.

Furthermore, independence from window size allows the AGPF-FastSLAM algorithm to more efficiently correct process model noise statistics for a wide spectrum of environmental disturbance, as compared to the AEKF-FastSLAM algorithm which relies on application-specific user-defined window sizes.
Chapter 8

Conclusions and Future Work

8.1 Conclusions

Simultaneous Localization and Mapping (SLAM) algorithms are the most basic layer of most robotic systems, on top of which other modules, such as exploration and path planning, are built. While a wide variety of SLAM algorithms have been formulated and successfully deployed, the emergence of new robotic applications and platforms have proved the current state of the art SLAM algorithms to be subpar to facing the challenges arising from the new scenarios.

All current state of the art SLAM algorithms suffer from a major drawback of the underlying sensor fusion algorithm, namely the assumption of a priori perfect knowledge of vehicle and observation model noise statistics. The robot is assumed to operate in a static environment lacking unmodelled noise. Sensor degradation due to wear-and-tear arising from continuous operation is not considered, neither is noise arising from unmodelled sources of error, such as environmental noise.

Wrongly estimated noise statistics or mismodeled interoceptive models can cause divergence in the state estimate, regardless of the sensor fusion algorithm used, as this thesis has
Conclusions and Future Work

shown. Consequently, the deployment of adaptive sensor fusion based SLAM algorithms, that adapt to changing model noise statistics, is necessitated. This thesis has used the example of an Autonomous Surface Craft (ASC) to successfully demonstrate this conclusion.

All state-of-the-art SLAM algorithms are built around a sensor fusion module, which is based on variations of the standard Kalman Filter in a majority of cases. Since the most frequently used algorithm for online estimation of Kalman Filter noise statistics is the Kalman Smoother-based expectation maximization (EM) algorithm, this thesis first used the Kalman Smoother as the basis of a formulation for an interoceptive algorithm to adaptively estimate the vehicle motion model noise covariance. While initially displaying promising results in simulation, further analysis of the Kalman Smoother revealed it to be a prey of the same drawbacks of the Extended Kalman Filter, namely a poor response to nonlinear models.

The Adaptive Limited Memory Filter (ALMF) was explored as an interoceptive model in response to the disadvantages of the Kalman Smoother. The Adaptive Extended Kalman Filter SLAM (AEKF-SLAM) was formulated similar to the Kalman Smoother-based algorithm to adaptively motion model covariance. The AEKF-SLAM formulation compared favorably to the Kalman Smoother SLAM formulation, as well as the standard EKF-SLAM formulation. The AEKF-SLAM was then used as an estimator for a marine dataset, and compared favourably with other state-of-the-art SLAM algorithms.

The ALMF was further extended to simultaneously estimate motion model bias and covariance in the particle filter based AEKF-FastSLAM algorithm. This was favourably benchmarked using the marine dataset against other state-of-the-art SLAM algorithms, including the previously formulated AEKF-SLAM algorithm.

The ALMF-based AEKF-SLAM and AEKF-FastSLAM algorithms were then comprehensively analysed using simulations, and were discovered to be overreliant on the size of the window size of the ALMF. Slight changes to the window sized resulted in a large change to the performance of the estimator, as analysed over a wide variety of scenarios.
In order to overcome this drawback, a Monte Carlo-based approach was chosen in favour of the window-based approach used previously, resulting in exploration of the Gaussian Particle Filter (GPF). Prediction density generated the GPF was favourably compared with prediction densities generated from similar state-of-the-art sensor fusion approaches, such as the EKF, Unscented Kalman Filter (UKF), Central Difference Kalman Filter (CDKF), using the Kolmogorov Smirnov statistic.

The GPF was then used as the basis of the sensor fusion module in the GPF-SLAM algorithm. The algorithm derived was compared favourably with other state of the art SLAM algorithms. The comparison was performed on the basis of simulation results, as well as the estimator results from the previously used marine dataset.

The strengths of the GPF led to it being formulated as an interoceptive algorithm for the FastSLAM algorithm, leading to the inception of the Adaptive Gaussian Particle Filter FastSLAM (AGPF-FastSLAM) algorithm. The APGF-FastSLAM algorithm was found to perform favourably in comparison of other SLAM algorithms, as well as the previously formulated AEKF-SLAM and AEKF-FastSLAM algorithms, without suffering from similar drawbacks due to overreliance on window size, as compared to the ALMF-based algorithms.

8.2 Future Work

Exploration of interoceptive model parameter estimation modules for SLAM resulted in the formulation and successful deployment of the AGPF-FastSLAM algorithm.

An encouraging approach currently being explored is the explicit calculation of the motion model noise statistics, with the aim of feeding them back to other modules of the autonomous robot system to improve operational efficiency of the platform. Initial attempts have been made in the direction of online estimation of motion model covariance, and favourable results have been obtained from simulations as seen in figs 8.1, 8.2, 8.3 and 8.4.
Conclusions and Future Work

Fig. 8.1 Estimation of angle standard deviation when initial model standard deviation is greater than actual standard deviation.

Fig. 8.2 Estimation of velocity standard deviation when initial model standard deviation is greater than actual standard deviation.
8.2 Future Work

Fig. 8.3 Estimation of angle standard deviation when initial model standard deviation is lesser than actual standard deviation.

Fig. 8.4 Estimation of velocity standard deviation when initial model standard deviation is lesser than actual standard deviation.
The algorithm formulation is based on stationary nature of the particle filter estimates. The simulated trajectory shown in fig.3.4 was used to evaluate the algorithm. A paper is currently in preparation with more detailed analyses being prepared from the initial algorithm.

A candidate for further improvement is the fixed number of normal distributions used by the current formulation of the AGPF-FastSLAM algorithm to adaptively estimate model parameters. A promising avenue of future research may be the incorporation of the Kullback-Liebler divergence metric to adaptively change the number of normal distribution, in response to change in noise statistics of the underlying system.

Other interesting avenues of research include formulation of the AGPF-FastSLAM algorithm with a Random Finite Set (RFS) based map model, in order to perform better in the presence of clutter in observations obtained from the exteroceptive sensor. The AGPF can also be deployed to estimate exteroceptive sensor parameters online, using proprioceptive sensor observations, as well as aposteriori estimates.
References


Appendix A

Derivation for optimal Covariance

Let $\hat{\lambda}$ be the set of parameters of the current stochastic model, and $\lambda$ represents a candidate parameter set.

The $Q$ objective function for maximizing expectation takes the following form:

$$Q(\hat{\lambda}, \lambda) = \int p(x_{0:N-1}|y_{0:N-1}, \hat{\lambda}) \log p(x_{0:N-1}|y_{0:N-1}, \lambda)$$  \hspace{1cm} (A.1)

Where, using Bayes’ rule and Markov Chain assumption,

$$\log p(x_{0:N-1}|y_{0:N-1}, \lambda) = \log p(x_1|\lambda) + \sum_{k=0}^{N-1} \log p(x_{k+1}|x_k, \lambda) + \sum_{k=0}^{N-1} \log p(y_k|x_k, \lambda)$$  \hspace{1cm} (A.2)

For ease of derivation, the state is assumed to be affected by noise $\sim \mathcal{N}(0, Q)$. Let $\eta_u$ be the inverse covariance (ie, $\eta_u = Q^{-1}$)

Now, consider the following properties:

$$\frac{\partial \log |A|}{\partial A} = (A')^{-1}$$ \hspace{1cm} (A.3)
Derivation for optimal Covariance

\[
\frac{\partial B'AB}{\partial A} = BB' \tag{A.4}
\]

Where \(A\) and \(B\) are two variables considered for the purpose of illustration.

Hence,

\[
2 \frac{\partial \log(p(x_{k+1}|x_k, \lambda))}{\partial \eta_u} = \frac{\partial \log|\eta_u|}{\partial \eta_u} - \frac{\partial (x_{k+1} - Fx_k)' \eta_u (x_{k+1} - Fx_k)'}{\eta_u} \tag{A.5}
\]

Where \(x_k\) is the state estimate at time step \(k\), and \(F\) is the transition matrix.

Simplifying, we obtain the following expression:

\[
\frac{\partial \log(p(x_{k+1}|x_k, \lambda))}{\partial \eta_u} = \frac{1}{2} (Q - (x_{k+1} - Fx_k)(x_{k+1} - Fx_k)') \tag{A.6}
\]

Hence,

\[
\frac{\log(p(x_{0:N-1}|y_{0:N-1}, \lambda)}{\partial \eta_u} = (N - 1) \frac{Q}{2} - \frac{1}{2} \sum_{k=0}^{N-1} (x_{k+1} - Fx_k)(x_{k+1} - Fx_k)' \tag{A.7}
\]

Equating the RHS of the above equation to zero for maxima, we obtain the expression for the optimal covariance as:

\[
Q_{EM} = \frac{1}{N-1} \sum_{k=0}^{N-1} (x_{k+1} - Fx_k)(x_{k+1} - Fx_k)' \tag{A.8}
\]
Appendix B

Publication List

B.1 Journal Papers


2. Akshay Rao; Wang Han, "A comparison of SLAM prediction densities using the Kolmogorov Smirnov Statistic," Journal of Unmanned Systems, To be published


B.2 Conference Papers

