UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Robust and Efficient Smoothing for Underwater Navigation

by Sajad Hassan

Thesis submitted in fulfilment of the requirements for the degree of

Master of Engineering (Research)

under the supervision of Professor Shoudong Huang, Professor Jonghyuk Kim and Professor Sarath Kodagoda

University of Technology Sydney
Faculty of Engineering and Information Technology

January 2023

Certificate of Original Authorship

I, Sajad Hassan declare that this thesis is submitted in fulfilment of the requirements for the award of Master of Engineering (Research) in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information resources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signed: Production Note: Signature removed prior to publication.

Date: 19 January 2023

Robust and Efficient Smoothing for Underwater Navigation

by

Sajad Hassan

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Engineering (Research)

Abstract

Computationally efficient and robust data fusion algorithms are valuable in navigation (or localisation) applications using low-cost sensors. This thesis considers the problem of enhancing the robustness and efficiency for 6-DoF (Degree of Freedom) underwater IMU (inertial measurement unit)-vision based navigation (or localisation). The emphasis is placed on algorithms that are robust and operate efficiently using low-cost inertial-visual sensors in underwater environment where it is vulnerable to outlier measurements. Such capability is desirable for autonomous underwater navigation.

One major factor that degrades the navigation accuracy are outlier measurements. In particular, inertial-visual underwater navigation is susceptible to wrong observation measurements. As a result, online and constant time robust state estimation techniques are valuable to provide the smoothed and enhanced vehicle trajectory. Existing solutions have mainly focused on increasing the robustness of the (extended) Kalman filter ((E)KF). They often require tuning the motion and prediction model noise covariance matrices that are fairly involved.

The contributions of this thesis arise from proposing a robust Biswas-Mahalanobis Fixed lag smoother (BMFLS) by utilising EKF, and a robust sliding window filter (RSWF) using the nonlinear least-squares (NLS) optimisation approach. The robust-BMFLS solution performs outlier rejection using the Chi-square test through reclassification and iterative smoothing. The limitation is it requires more iteration in time intervals with high ratio of outliers.

While, in the NLS optimisation approach works by assigning a weight to each observation, that are iteratively computed from the robot pose prediction error and observation error, outliers are detected and rejected by classification expectation-maximisation. However, solving the NLS optimisation using full-batch estimation is an offline process. By introducing the RSWF, a constant time and online solution is presented. This is an incremental and online robust solution which is computationally efficient to robust full-batch estimation. The impact of different optimisation window sizes and update periods are studied on the navigation performance. This is useful to determine the optimum window size and update period using RSWF for localisation.

Acknowledgements

First and foremost I thank God for everything he has given me in this life especially my parents whom have been always supportive and caring for my learning.

I would like to thank my supervisory team, Prof. Jonghyuk Kim and Prof. Shoudong Huang for your guidance, technical support and patience during my candidature, especially for your tireless and thorough reviews. I would also like to thank my co-supervisor, Prof. Sarath Kodagoda for his support.

I would also like to thank Dr. Andrew To for replying to my Emails. Equally, I acknowledge Dr. Andrew To and Dr. Khoa Le for collecting the experimental data. Also to everyone in the weekly SLAM meetings, and a huge thanks to my work colleagues for being patient and supportive for the past two years.

The experiences throughout this degree has been different with COVID and mix of working from home and uni. Robotics has been something new for me which I did not know much about before. But I have a much broader view about it now and its different disciplines.

Finally, I would like thank the school of MME and Faculty of Engineering and Information Technology and UTS for their support and the conference fund.

Contents

List of Figures

List of Tables

Acronyms & Abbreviations

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
AUD	Australian Dollar
BMFLS	Biswas-Mahalanabis Fixed-Lag Smoother
CAS	Centre for Autonomous Systems
DoF	Degree of Freedom
DCM	Direction Cosine Matrix
FOGM	First Order Gauss-Markov
EKF	Extended Kalman Filter
EM	Expectation-Maximisation
IMU	Inertial Measurement Unit
IRLS	Iterarively Re-weighted Least-Squares
KF	Kalman Filter
MAP	Maximum a Posterior
MD	Mahalanobis Distance

NLS Nonlinear Least-Squares

PF Particle Filter

RMSE Root-Mean Square-Error

ROV Remotely Operated Underwater Vehicle

ROS Robot Operating System

SIS Sequential Importance Sampling

SIR Sequential Importance Resampling

SLAM Simultaneous Localisation and Mapping

RSWF Robust Sliding Window Filter

SPIR Submersible Pile Inspection Robot

SWF Sliding Window Filter

UTS University of Technology, Sydney

Nomenclature

	General Notations
$\mathbf{b}_k$	IMU bias state at time step k
d_k	Mahalanobis distance time step k
$\mathbf{f}(\cdot)$	Dynamic transition function in state space model
$\mathbf{F}$	State transitional matrix
$\mathbf{F}_{ins}$	Main states transitional matrix
$\mathbf{g}(\cdot)$	Coupling of random process noise function in dynamic model
$\mathbf{g^n}$	Earth's gravitational acceleration
$\mathbf{G}$	Coupling matrix between random process noise and state of dynamic model
$\mathbf{H}$	Observation model matrix
$\mathbf{k}$	Time step number
$\mathbf{K}$	Kalman gain
$p(\cdot)$	Probability density function
$\mathbf{L_v}$	Prediction model squared error
$\mathbf{L_z}$	Observation model squared error
$\hat{\mathbf{N}}_{\mathbf{eff}}$	Effective sample size
$\mathbf{Q}$	Covariance of the process noise
$\mathbf{P}_{k\|k}$	Covariance of EKF/Gaussian filter at time step k
$\mathbf{R}$	Covariance of the observation noise
$\mathbf{S}$	Innovation covariance matrix
$\mathbf{x}_k$	State at time step k
$\mathbf{x}_v$	Vehicle state

$\mathbf{u_k}$	IMU input at time step k
$\mathbf{v}$	Observation noise
$\mathbf{w}$	Process noise
$\mathbf{z}_k$	Observation measurement at time step k
Δt	Sampling period
ν	Observation innovation

Chapter 1

Introduction

1.1 Background and Motivation

Robust navigation is crucial for underwater robots to know its pose (position and orientation) in a global frame. This is useful in making the robot follow a desired trajectory, or performing a task such as cleaning or manipulation in intervention missions. Remotely operated underwater vehicles (ROVs) have played a critical role in carrying out tasks considered difficult or dangerous for humans. Unlike surveillance and inspection tasks, intervention missions have higher demand on the navigation accuracy [1] [2]. For instance Chavez et al. [3] admit underwater localisation is challenging in their work. In particular, when the robot is close to the object to be manipulated [4]. But knowing the robot pose in the global frame is essential in achieving these goals.

The underwater domain is one of the hostile and challenging environment for ROV to operate. While many land-based or aerial navigation rely on one or more global positioning system (GPS) to obtain an accurate measurement of vehicle's position, a robot operating underwater typically does not have this information [5]. Often low visibility and poor quality sensor data or sensor failures make the vehicle underwater navigation tracking difficult. This results in wrong pose observations (measurements) that introduce outliers which degrade the navigation accuracy (Figure 1.1). Outlier detection is inherently a difficult task which along with smoothing the navigation trajectory tend to increase the

computational requirements. Further, the problem becomes more challenging using low-cost visual inertial sensors. Low-cost here is considered inertial-visual sensors where the IMU costs within few thousand Australian Dollars (AUD) (about low-quality tactical grade (<3K AUD), while the camera (vision) is less than 500 AUD.

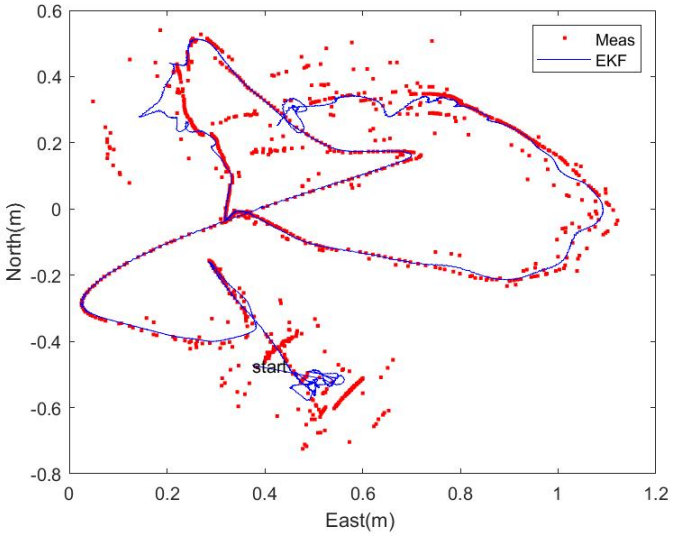

FIGURE 1.1: An EKF estimated trajectory projected in 2D, using the experimental dataset (red dots are observation measurements)

FIGURE 1.2: The submsersible pile inspection robot version 3 (SPIR3) ROV.

1.2 Objectives

The objective of this thesis is to develop robust and efficient smoothing solutions for 6-DoF navigation using low-cost IMU-vision sensors. They are:

- A robust Biswas-Mahalanabis Fixed-Lag Smoother (BMFLS) using extended Kalman filter (EKF),
- An efficient and robust sliding window filter (RSWF) solution using nonlinear least-squares optimisation.

Here, the RSWF is an incremental solution which works by scaling down the full batch method to a small window of optimisation to remove potential outliers and perform state estimation, and then moving to the next window. The developed solution are verified using real-world experimental dataset in a water tank environment using the submersible pile inspection robot version 3 (SPIR3) as shown in Figure 1.2. Although the outcome of this work could be applied to other navigation problems, this research addresses unique solutions to address low-cost IMU-vision based underwater localisation using fiducial markers for pose measurement, to achieve robust inertial-visual navigation solution.

1.3 Contributions

The main contributions of this thesis are:

- Developing an efficient RSWF navigation solution with expectation-maximisation (EM) using optimisation;
- Investigating the impact of various window sizes and update rate on accuracy and computation run-time for RSWF estimation given unknown dataset;
- Enhancing ARTag underwater navigation performance using low-cost inertial-visual sensors;
- Developing an iterative smoothing and outlier detection by utilising BMFLS.

1.4 Publications

S. Hassan, H. Byun, J. Kim, "Iterative Smoothing and Outlier Detection for Underwater Navigation" in *Australasian Conference on Robotics and Automation (ACRA)* 2021.

S. Hassan, J. Kim, S. Huang, "An Incremental Robust Underwater Navigation with Expectation-Maximisation" in *Australasian Conference on Robotics and Automation (ACRA)* 2022.

1.5 Thesis Structure

The summary breakdown for this thesis are as follows:

Chapter 2 provides a background on state estimation techniques, outlier detection methods, a literature on the existing work to enhance filtering performance (robustness and efficiency) of state estimation techniques and a discussion on their shortcomings.

Chapter 3 introduces inertial navigation and provides the mathematical derivation of the inertial navigation equations, description of the experimental environment, and the nonlinear process model and the observation model for this thesis problem.

Chapter 4 presents the iterative smoothing based outlier detection approach using the BM-FLS and the Chi-Square statistical test, the results achieved and evaluates the performance of this EKF based approach.

While, Chapter 5 presents the RSWF navigation solution using nonlinear least-squares, the results achieved, a discussion about the performance of this method and a comparison to the approach in Chapter 4.

Finally, Chapter 6 draws a summary of this work and conclusions, some recommendations with possible future investigations.

Chapter 2

Literature Review

2.1 Overview

This chapter provides a review of literature related to navigation tracking. They are: the filtering and smoothing techniques with a discussion on the performance (accuracy and efficiency), outlier detection methodologies, bias estimation, and work carried out so far to increase the robustness and efficiency of state estimators and their shortcomings.

2.2 State Estimation Techniques

In the past, sailors used various tools such as compass, sexant, astrolabe or quadrant in marine navigation. But modern navigation has become a combination of science and technology. State estimation in vehicle navigation mainly involves predicting the position, velocity and orientation, which if known, allows to understand the motion of the vehicle over time. There are number of state estimation techniques that could be used in navigation problems.

The full-least squares optimisation technique and the Kalman filter or extended Kalman filter (EKF) are the most widely used state estimation techniques which are discussed in the next section.

2.2.1 Full Least-Squares

The full least-squares optimisation was one of the early optimal estimation methods and it is still much in use today [6]. Optimisation has several advantages compared to filtering techniques such as EKF. The nonlinear least-squares (NLS) estimation optimises over entire trajectory by using entire measurements to perform full batch-estimation (Figure 2.1). As a result, optimisation provides smoothing which much desired to filtering, whereas the EKF is sub-optimal compared to full-batch estimation. In [7] Sjanic et al. present a detailed approach of the nonlinear least-squares framework via Gauss-Newton and argue EKF suffers from linearisation errors.

However, solving the NLS by full-batch estimation in an offline process which becomes computationally expensive with growing number of states. This is not desirable for real-time applications. Further, NLS is known to be sensitive to data contamination. As a result, the navigation accuracy will degrade measurement observations are contaminated with outliers.

2.2.2 Kalman Filter

Kalman filter (KF) is the optimal estimator for linear models with Gaussian noise. that recursively estimates the states. Nevertheless, its performance can be impaired by outliers in the measurements. The KF algorithm is not shown here but runs similar to EKF as outlined in the next section.

2.2.3 Extended Kalman Filter

The EKF is an extension of the Kalman filter to nonlinear problems. It is widely used in navigation applications due its simplicity and efficiency. The EKF assumes the Markov property. That is, the prior of k^{th} state x_k is only related to x_{k-1} and is independent of the states before $k-1$ ([8]). That is:

$$p(\mathbf{x_k}|\mathbf{x_{1:k-1}}, \mathbf{z_{1:k-1}}) = p(\mathbf{x_k}|\mathbf{x_{k-1}}) \tag{2.1}$$

Also, the current observation $\mathbf{z}_k$ given the current state $\mathbf{x}_k$ is conditionally independent of the past measurements and states. That is:

$$p(\mathbf{z_k}|\mathbf{x_{1:k}}, \mathbf{z_{1:k-1}}) = p(\mathbf{z_k}|\mathbf{x_k}) \tag{2.2}$$

In summary, the EKF uses the previous state $\mathbf{x}_{k-1}$ and current observation $\mathbf{z_k}$ to estimate $\mathbf{x_k}$, unlike the optimisation approach in Section 2.2.1 that uses all the historical data. Further, the EKF is also vulnerable to outliers [9]. The Equation (2.3) summarises the EKF progression,

$$\begin{aligned}
\mathbf{x_{k|k-1}} &= \mathbf{f}(\mathbf{x_{k-1|k-1}}, \mathbf{u_k}) \\
\mathbf{P_{k|k-1}} &= \mathbf{F_k}\mathbf{P_{k-1|k-1}}\mathbf{F_k}^T + \mathbf{Q_k} \\
\mathbf{e_k} &= \mathbf{z_k} - \mathbf{H_k}\mathbf{x_{k|k-1}} \\
\mathbf{K_k} &= \mathbf{P_{k|k-1}}\mathbf{H_k}^T(\mathbf{H}_k\mathbf{P_{k|k-1}}\mathbf{H_k}^T + \mathbf{R}_k)^{-1} \\
\mathbf{x_{k|k}} &= \mathbf{x_{k|k-1}} + \mathbf{K_k}\mathbf{e_k} \\
\mathbf{P_{k|k}} &= (\mathbf{I} - \mathbf{K_k}\mathbf{H_k})\mathbf{P_{k|k-1}}
\end{aligned} \tag{2.3}$$

where

- $\mathbf{x}_{k|k-1}$ and $\mathbf{x}_{k|k}$ are the prior and update state predictions respectively,
- $\mathbf{f}(\cdot)$ is the nonlinear process model function,
- $\mathbf{u}_k$ is the IMU input
- $\mathbf{F}_k$ and $\mathbf{H}_k$ are the process and observation model jacobians matrices,
- $\mathbf{P_{k|k-1}}$ and $\mathbf{P_{k|k}}$ are the prior and updated covariance matrices respectively,
- $\mathbf{Q_k}$ and $\mathbf{R_k}$ are the covariance process and observation noise matrices,
- $\mathbf{K}_k$ is the Kalman gain,
- $\mathbf{z}_k$ is observation measurement, and
- $\mathbf{e}_k$ is the observation error.

2.2.4 Particle Filter

Particle filters (PFs) are also sub-optimal filters. The PF technique seeks to estimate the probability density function (pdf) by obtaining by a sample of particles and their corresponding weights. The states are then estimated based on the sample and the associated weights. The sequential importance sampling (SIS) filter is the basis of PF, however, it suffers from degeneracy problem. Instead, the sequential importance resampling (SIR) PF is introduced. There are also other types PF derived from SIS filter, but SIR is investigated in this thesis. Because the SIR filter is explored without knowledge of the observation makes it inefficient and prone to outliers [10].

However, some variants of the PF are capable of handling nonlinear process and observation model, and non-Gaussian noises. Lastly, the PF outperforms EKF but it becomes numerically challenging for high-dimensional state estimation problems with growing number of samples [11].

2.3 Optimal Smoothing

Smoothing provides more accurate state estimation compared to filtering methods (i.e EKF). There are various type of smoothers, fixed-interval smoother, fixed-lag smoothers (FLS) and fixed-point smoothers. Fixed lag smoothers can operate online but estimates the vehicle state at a delayed time [12]. On the other hand, fixed-interval smoother uses all the measurements in the interval, and commonly used for post-processing [13].

Although there are various typers smoothers, some may not be practical. The important performance criteria for optimal smootheers are numerical stability, computational complexity and memory requirements.

2.3.1 Biwas-Mahalanabis Fixed-Lag Smoother

The BMFLS is a fixed-lag smoother (FLS) generating estimation at time step $k - l$ using the current measurement at time step k. It is essentially the KF or EKF with augmented state vector from state estimation over discrete-time window of fixed size l.

The BMFLS runs in real time but estimates the state in delayed time. The earliest types of fixed-lag smoothers were found to suffer from numerical stability. The BMFLS first discovered by Biswas and Mahalanabis [13] is rather simple and numerically stable. The detail implementation of the BMFLS is detailed in Section 4.2.

2.3.2 Sliding Window Filter

For the nonlinear least-squares optimisation to operate efficiently, the state vector cannot grow without bound. One simple way to maintain this is to remove old poses. The Sliding window filter (SWF) was first introduced in [14] to perform constant time online state estimation for optimisation approaches such as NLS. The SWF iterates over a window of n steps and provides smoothed estimation. The window of optimisation is then moved forward and carries on (Figure 2.1). However, removing directly the oldest parameters from system solution results in information loss and gives sub-optimal solution. The correct way to remove old parameters is to perform marginalisation using Schur complement to take into account prior information [15].

While the SWF does ease the computational burden, but from optimisation point of perspective is not significant and as a result SWF are still an active area of research [11].

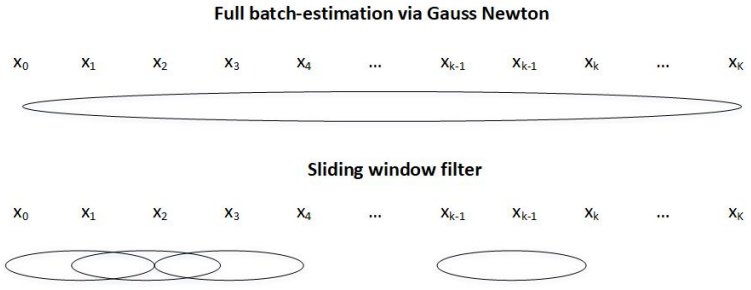

FIGURE 2.1: An illustration of full-batch estimation and sliding window filter. The full-batch estimation iterates over entire trajectory, whereas SWF iterates over a window of steps.

2.3.3 Incremental Smoothing

The concept of incremental smoothing was introduced by Kaess et al. [16] in the SLAM problem to deal with computational complexity involved with full-batch estimation. While, the SWF performs updating of all states in the sliding window, this may not be always required as some states remain unchanged in certain conditions. As a result the notion proposed by Kaess et al. does periodic updates of every 10 or 100 steps to improve the efficiency of the incremental SLAM.

A similar concept introduced in [17] called compressed-SLAM was to deal with computational requirements. In the compressed-SLAM, the states are partitioned into local map and global map. In the compressed-SLAM the local map are actively updated. While the global landmarks are updated at a much lower rate thus compressing the local-to-global correlation information [17]. The main point here is, states that remain close to constant can have low cycle update to improve the filtering efficiency while maintaining accuracy.

2.4 Outlier Detection

Outlier detection mechanism is critical in vehicle navigation applications. In particular, when using low-cost sensors to navigate in a dynamic environment that makes the platform vulnerable to outlier measurements. "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by different mechanism" [18]. Outliers may occur due to noisy sensor measurements, sensor failures or environmental disturbances [19]. Throughout this thesis it is evident that in underwater IMU-vision based navigation using ARTag fiducial markers to obtain the pose measurements are prone to outliers.

Further, capturing outliers is inherently difficult. Ignoring outliers will deteriorate the navigation accuracy and as a result robust algorithms are essential. This section discusses the performance of different outlier detection techniques and the existing work.

2.4.1 Chi-square Test

The Chi-square test or Mahalanobis gating test is a well known technique to screen-out outlier observations in navigation applications [20][21][22] [23]. It is commonly applied in Kalman filtering to detect and rejects outliers by calculating the Mahalanobis distance (MD) d_k,

$$d_k = \nu_k^T S_k \nu_k. \tag{2.4}$$

The Chi-square $\boldsymbol{\chi}^2$ distribution curve is determined by degree of freedom associated with the distribution [24]. In Kalman filtering, if the dimension of measurements is m, using Chi-square distribution with m degrees of freedom $\boldsymbol{\chi}^2(m)$ is used to classify the measurement using the criterion:

$$\begin{cases} \text{if } d_k < \chi_T^2, \text{ Inlier} \\ \text{if } d_k \geq \chi_T^2, \text{ Outlier} \end{cases} \tag{2.5}$$

where χ_T^2 is the threshold value from $\boldsymbol{\chi}^2(m)$ for a given percentile.

However, Mahalanobis gating test may fail with high intensity of outliers which may result in rejecting correct observations causing information loss. An example of this is could be seen in Chapter 4, in the experimental results. The direct catastrophic consequence is we may lose all the good information.

2.4.2 M-estimation

As previously mentioned, the least-squares is sensitive to outliers. The maximum likelihood M-estimation seeks to reduce the least-squares sensitivity by replacing with a more robust cost function such as Cauchy or German-McClure. The weight function produced by these cost functions means large errors will not carry as much weight and have less influence on the estimation.

2.4.3 Iteratively Re-weighted Least-squares

The iteratively re-weighted least-squares (IRLS) also allows to reduce the influence of large errors on the estimation by re-evaluating the prediction and observation error in the context of the navigation problem. M-estimation and IRLS do not eliminate an outlier measurement but rather reduce its influence.

2.4.4 Expectation-Maximisation

The expectation-maximisation (EM) algorithm is an iterative method to find the maximum-likelihood (ML) estimate of parameters of underlying distribution that cannot be solved directly [8]. It has applications in EKF [25] to tune the noise covariance parameters or in SLAM [26] to ease the computational complexity. The EM algorithm iterates between the expectation (E) step and maximisation (M) step. After which the distribution of the latent variables in the E step is determined. In this thesis, the EM algorithm is applied to NLS batch-estimation to identify outliers.

2.5 Bias Estimation

Sensor white noise and bias instability are two main contributors in noisy measurements. Accounting for bias in inertial navigation estimation improves the navigation accuracy by constraining the drift during each IMU cycle and during loss of visual camera. As a result, it helps in smoothing out trajectory by reducing the drift of IMU sensors and improved state estimation during visual camera failures [27].

The Gauss-Markov model is effective than the random constant and random walk models for bias modelling. In particular, the first order Gauss-Markov (FOGM) model is considered suitable for bias modelling in sequential navigation filters [28]. The FOGM integration is shown as below:

$$\mathbf{b}_{k+1} = \mathrm{e}^{-\Delta t/\tau}\mathbf{b}_k + \mathbf{w_k} \tag{2.6}$$

where τ denotes time-constant and $\mathbf{b_k}$ is the bias state at time step k.

2.6 Related Work on Outliers

There are various ways that researchers have studied to mitigate the effect of outliers on state estimation. Ting et al. [29] introduces a scalar weight for each measurement according to Gamma distribution. By setting up an EM framework, the KF parameters, process and observation model noise covariance matrices ($\mathbf{Q}$ and $\mathbf{R}$) are iterated until convergence.

Instead Agamennon et al. [19] models the observation noise matrix using the Student t-distribution that is allowed to vary over time that may have tails compared to Gaussian distribution. The expected measurement noise covariance matrix is found by applying EM algorithm. They further apply fixed-interval smoothing to enhance the estimation. Finally, the ($\mathbf{Q}$ and $\mathbf{R}$) are still required to be estimated.

While, a robust EKF is introduced in [21] where the Rauch-Tung-Striebel (RTS) is applied as fixed lag smoother. Similarly, this method also requires the estimation of ($\mathbf{Q}$ and $\mathbf{R}$). But learning the process and observation model noise covariance matrices are fairly involved process. Lee et al. [23] extended the work in [19] by performing the Chi-Squared statistical test to check it's outlier measurement and remove it. Otherwise, it moves with method in [19] for the expected observation noise covariance matrix. But, their assumption is "outliers don't always arise" and the approach is sufficient for the work.

Moreover, a robust KF is introduced in [20] based on MD as the judging criterion. Suppose the MD is greater than the Chi-square distribution, a scaling factor is introduced to rescale the innovation covariance that reduces the Kalman gain to maintain robustness. However, this approach is applicable for low-dimensional problems.

Furthermore, the majority of the stated work were on enhancing the robustness of the KF [21]. Many have not considered the nonlinear least-squares optimisation method. Lee et al [30] has demonstrated promising results using the full least-squares approach to detect and eliminate pose-graph false loop closures based on the classification EM method. In this

work, a weight is assigned to each measurement Cauchy weight function that are iteratively computed from the errors between the predicted poses and observed measurements. and solved via Gauss-Newton. However, this method runs offline and is computationally heavy for high-dimensional problems and long trajectories. In addition, Cheng et al. [31] applied the concept from [30] and proposed an efficient and robust linear pose-graph based SLAM to eliminate false loop closures by introducing a delayed optimisation approach.

The assumption in the above work is that the odometry solutions are reasonably accurate. But, if low-cost inertial-measurements are used, this assumption may be no longer valid. Also, the work above were simulation based or validated for aerial or land navigation and not for difficult environments such as underwater where due light absorption and scattering through water make underwater vision based localisation challenging.

On the other hand, Peng et al. [32] have formulated a robust PF for underwater terrain-aided navigation. The approach modifies the covariance matrix in the likelihood function $p(y|x)$ based on the Huber cost weight function to suppress the effect from outliers. The notion in Chapter 5 can be expanded to the solution in [32] to develop a further robust PF.

Finally, fiducial markers are used in vision-based navigation for pose estimation. Fiducial markers have several advantages. First, their tone and shape are easy to recognise that are printed in black and white which is easy to differentiate in a binary image. Second, they are low-cost and are installed to planar surfaces (e.g. floors, ceilings) with known global coordinates [33]. There are several fiducial marker packages available (e.g. ARTag, AprilTag, ArUco, STag). In an evaluation study for these four different packages by Kalaitzakis et al. have indicated ARTag have the lowest computational cost, but suffer from extreme outliers in marker bundles [34].

2.7 Summary

This chapter provided some background in state estimation techniques and outlier detection methods where a critical review of the performance of these techniques were discussed. Also, a literature review on the existing work on enhancing the robustness of the KF, EKF

and full-least squares were presented and became clear they are computationally challenging. Finally, from literature review it become evident the work towards enhancing the navigation algorithms for underwater visual-inertial applications are lacking and requires more consideration.

Chapter 3

Problem Formulation

3.1 Problem Overview

This thesis is concerned with enhancing robustness of IMU-vision based navigation using low-cost inertial-visual sensors in a 6-DoF environment where a monocular camera is used to obtain the robot pose measurement using AR (augmented reality) Tag fiducial markers. The details of the experimental setup is outlined in Section 3.3.

However, due to harsh underwater environment, tracking an underwater robot becomes difficult. In this thesis, the dataset collected from the experiment suffers from noisy and missing measurements due outlier presence in the underwater environment. Figure 3.1 illustrates an interval of experimental raw vision measurements for the x-position. It can be observed in the middle interval, there are three layers of observations where the top two layers appear as outliers. As a result, such measurement observations will lead to spurious navigation. The rest of this chapter provides the derivations for the inertial navigation equations, the process and observation model for the navigation problem described in this thesis.

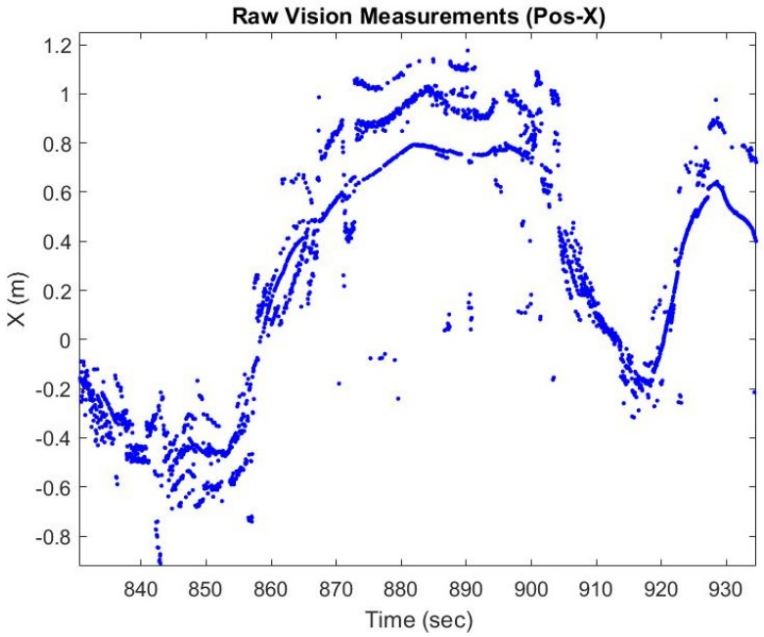

FIGURE 3.1: An example of outliers in the raw vision and marker observations. It is evident the noises do not follow normal standard Gaussian distribution, but rather an offset outlier pattern which stems from confusion in recognising the markers.

3.2 Inertial Navigation

Inertial navigation is a pose estimation technique in which the measurements provided by accelerometer and gyroscope are integrated to predict the vehicle's position and orientation from known starting point and orientation [35]. Inertial measurement units (IMUs) usually contain three orthogonal rate-gyroscope and three orthogonal accelerometers. The gyroscope measures angular velocity around each axis x, y and z while the accelerometer measures the acceleration along each axis x, y and z in the body frame. Figure 3.2 is the the Orientus IMU mounted onto the SPIR platform. The inertial navigation equations in this thesis are outlined in Section 3.2.2.

The SPIR platform's IMU noise characteristics are displayed in Table 3.1.

TABLE 3.1: The IMU noise characteristics.

Sensor	Accelerometer	Gyro
Bias Instability	20 μg	3 $^\circ/hr$
Initial Bias	$< 5\ mg$	$< 0.2\ ^\circ/hr$
Noise Density	100 $\mu g/\sqrt{Hz}$	0.004 $^\circ/s/\sqrt{Hz}$

FIGURE 3.2: The Oreintus IMU from Advanced Navigation, used in SPIR platform to provide 6-DoF vehicle information.

3.2.1 Coordinate Systems

There are various coordinate systems used in inertial navigation based on the applications and sensors. Because the sensor measurements and navigation outputs are expressed in different reference frames, the coordinate system must be defined to formulate the navigation equations. This section defines the references frames used for the navigation problem in this thesis.

Body Frame

The body frame (moving platform) consists of the three orthogonal axes with the origin being the centre of mass of the platform. The x-axis (or roll axis) points forward, the y-axis (or pitch axis) points to the right, and z-axis (or yaw axis) points down all with respect of platform that makes up the right-handed coordinate system.

Navigation Frame

The navigation frame is defined to be a non-accelerating reference frame. The origin of this coordinate system could be fixed anywhere in the space with the three orthogonal axes.

Camera Frame

Camera frame (c), is a moving with its origin fixed in the camera's optical centre. The axes of the camera frame is denoted as X_c, Y_c and Z_c where the X_c is the optical axis.

3.2.2 Inertial Navigation Equations

In this section, the attitude equations will be presented first followed by the velocity/position equations.

3.2.2.1 Attitude Equations

The Euler angle and the Direction Cosie Matrix (DCM) transformation will be presented.

Euler angles

In this thesis, the navigation frame is rotated and aligned to the body frame in the sequence of yaw (ψ), pitch (θ) and roll (ϕ). The transformation of the navigation frame to body frame $\mathbf{C}_n^b$ is constructed by multiplying each consecutive rotation matrices in the sequence. By taking the inverse of $\mathbf{C}_n^b$, the transformation of the body frame to the navigation $\mathbf{C}_b^n$ is obtained:

$$\mathbf{C}_b^n = \begin{bmatrix} C_\theta C_\psi & C_\phi S_\psi + S_\phi S_\theta C_\psi & S_\phi S_\psi - C_\phi C_\theta C_\psi \\ -C_\theta S_\psi & C_\phi C_\psi - S_\phi S_\theta S_\psi & S_\phi C_\psi + C_\phi S_\theta S_\psi \\ S_\theta & -S_\phi C_\theta & C_\phi C_\theta \end{bmatrix}, \tag{3.1}$$

where $S_{(\cdot)}$, $C_{(\cdot)}$ are shorthand notations for $\sin(\cdot)$ and $\cos(\cdot)$ respectively. As the body frame rotates with respect to the navigation frame, the Euler angles also change according to the angular rates measured by the gyro onboard the vehicle.

$$\begin{aligned}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} &= \begin{bmatrix}\dot{\phi}\\ 0\\ 0\end{bmatrix} + \begin{bmatrix}1 & 0 & 0\\ 0 & C_\phi & S_\phi\\ 0 & -S_\phi & C_\phi\end{bmatrix}\begin{bmatrix}0\\ \dot{\theta}\\ 0\end{bmatrix}\\ &+ \begin{bmatrix}1 & 0 & 0\\ 0 & C_\phi & S_\phi\\ 0 & -S_\phi & C_\phi\end{bmatrix}\begin{bmatrix}C_\theta & 0 & -S_\theta\\ 0 & 1 & 0\\ S_\theta & 0 & C_\phi\end{bmatrix}\begin{bmatrix}0\\ 0\\ \dot{\psi}\end{bmatrix}\\ &= \begin{bmatrix}1 & 0 & -S_\theta\\ 0 & C_\phi & S_\phi C_\theta\\ 0 & -S_\phi & C_\phi C_\theta\end{bmatrix}\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}.\end{aligned} \tag{3.2}$$

The inverse of Equation (3.2) is the Euler rates, that is:

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \mathbf{E}_b^n\boldsymbol{\omega} = \begin{bmatrix}1 & S_\phi S_\theta/C_\theta & C_\phi S_\theta/C_\theta\\ 0 & C_\phi & -S_\phi\\ 0 & S_\phi/C_\theta & C_\phi/C_\theta\end{bmatrix}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}. \tag{3.3}$$

Equation (3.3) shows the nonlinear transformation of the angular rates to the Euler rates. By implementing a first order numerical integration, the Euler angles can be expressed in discrete form, that is,

$$\begin{bmatrix}\phi_k\\ \theta_k\\ \psi_k\end{bmatrix} = \begin{bmatrix}\phi_{k-1}\\ \theta_{k-1}\\ \psi_{k-1}\end{bmatrix} + \int_{k-1}^{k}\begin{bmatrix}\dot{\phi}_{k-1}\\ \dot{\theta}_{k-1}\\ \dot{\psi}_{k-1}\end{bmatrix} dt. \tag{3.4}$$

The general form Equation (3.4) is expressed in this thesis as:

$$\boldsymbol{\psi}_{k+1}^n = \boldsymbol{\psi}_k^n + \mathbf{E}_b^n\boldsymbol{\omega}_k^b\Delta t, \tag{3.5}$$

where subscript b and superscript n denote body and navigation frame respectively.

3.2.2.2 Position and Velocity Equations

The body frame and navigation frame are illustrated in Figure 3.3. Integrating the IMU acceleration output $\mathbf{f_k^b}$ by time step $\mathbf{\Delta t}$ yields the body frame velocity. Using the DCM matrix, the body velocity with respect to (w.r.t) navigation frame is:

$$\mathbf{v}_{k+1}^n = \mathbf{v}_k^n + (\mathbf{C}_b^n[\mathbf{f}_k^b] + \mathbf{g}^n)\mathbf{\Delta t}. \tag{3.6}$$

Then, integrating the velocity w.r.t. fixed frame gives the position prediction, that is:

$$\mathbf{p}_{k+1}^n = \mathbf{p}_k^n + \mathbf{v}_k^n \mathbf{\Delta t}. \tag{3.7}$$

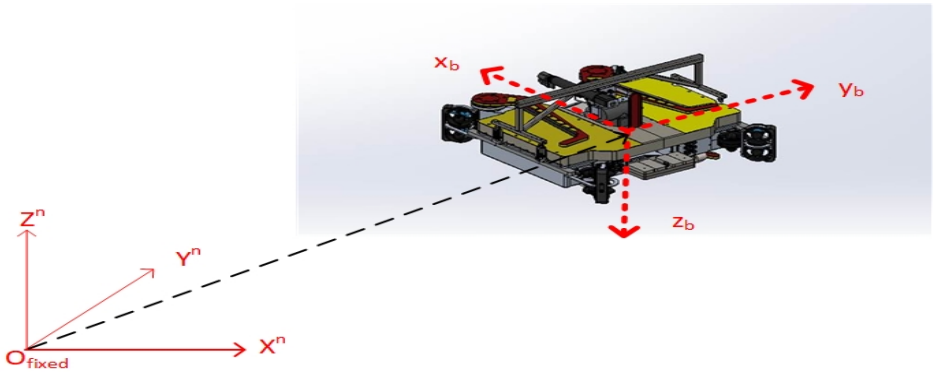

FIGURE 3.3: The navigation frame (n) and body frame (b) illustration with SPIR.

3.3 Environment Setup

This section describes the experimental setup to collect the dataset. This includes physical description of water tank facility, the monocular vision and the ARTag fiducial markers setup to obtain the pose measurements of the vehicle.

3.3.1 Experimental Environment

The experiment was set up inside a $(6 \times 4 \times 2m)$ water tank (Figure 3.4) at UTS infrastructure lab facility. Plain water from the tap with Hydrogen Peroxide is used to fill the tank. The SPIR (vehicle) operates in 6-DoF environment and hence position and orientation observation are required. The mechanism to obtain vehicle pose measurements is outlined in next section.

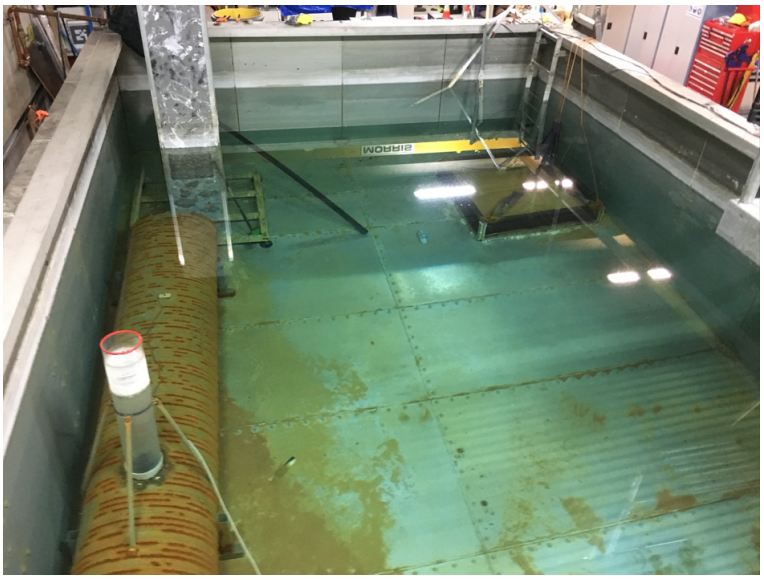

FIGURE 3.4: A general view of the UTS water tank facility.

3.3.2 Pose Measurements

Fiducial markers are often used in augmented reality to estimate camera pose [33]. Here, obtaining the vehicle position and orientation measurements requires estimating the camera pose. In the experiment, ARtag fiducial markers are installed inside the water tank (environment) as shown in Figure 3.5 with known global coordinates to provide the direct position and orientation measurements of the vehicle. The ARtag fiducial markers are printed in black and white using the *ar_track_alvar* ROS package. The markers are adhered to an acrylic sheet. The magnetically acrylic sheets were attached to one side of tank wall. An outline of obtain pose measurements using monocular vision (camera) is provided in the next section.

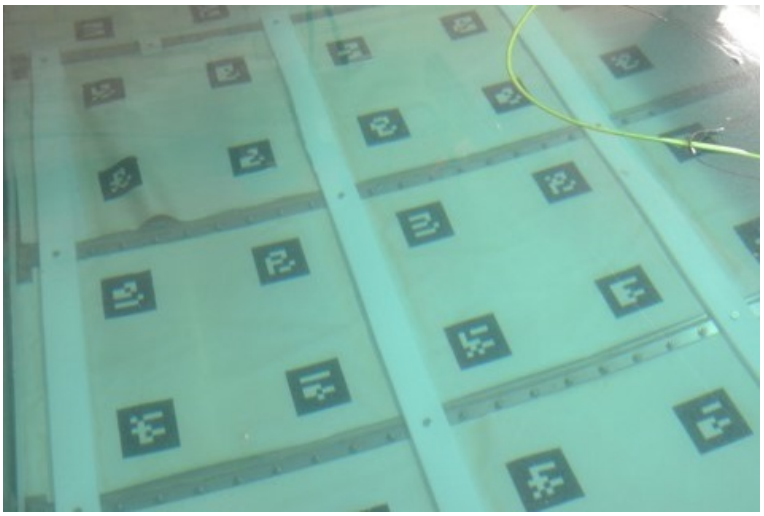

FIGURE 3.5: ARTag-based fiducial markers installed on the side wall of the water tank, which is used to provide the pose (position and orientation) measurements using a monocular camera installed on the robot. The markers' positions and orientation are pre-calibrated.

3.3.3 Vision

The monocular camera shown in Figure 3.6 is mounted inside an enclosure on the front of the platform. It takes 26 camera images in a second (26 Hz). The monocular camera is calibrated to know its intrinsic parameters (focal point and principal point). The intrinsic properties allow to convert the coordinates in the image to (scale-less) position in the world [36]. By knowing the camera's intrinsic and fiducial marker physical size, 6-DoF tracking is achieved. As a result, the environment observation setup provides the direct position and orientation visual measurements of the robot.

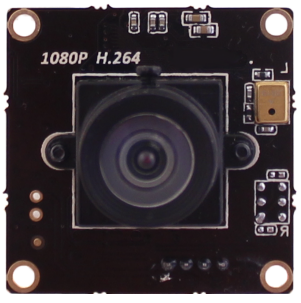

FIGURE 3.6: The monocular camera (Low-Light HD USB) camera mounted on the platform.

3.4 State Estimation Model

This section formulates the state estimation model for the underwater navigation problem in this thesis. This includes the derivation of the process and observation models.

3.4.1 Vehicle State

The 6-DoF vehicle state $\mathbf{x_k}$ is defined as the position, velocity and orientation in the navigation frame:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{p}_k^n & \mathbf{v}_k^n & \boldsymbol{\psi}_k^n \end{bmatrix}^T, \tag{3.8}$$

where $\boldsymbol{\psi}^n$ represents the Euler angles: roll (ϕ), pitch (θ) and yaw (ψ).

3.4.2 Nonlinear Process Model

The vehicle nonlinear process model can be expressed in the general form:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{g}(\mathbf{x}_{k-1}, \mathbf{w}_k), \tag{3.9}$$

where $\mathbf{x}_k$ is the vehicle state, $\mathbf{f}(\cdot,\cdot)$ represents the nonlinear state transition function at time k based on the previous time step state $\mathbf{x}_{k-1}$ and current control input (IMU output) $\mathbf{u}_k$. Also $\mathbf{g}(\cdot,\cdot)$ is the nonlinear coupling function between random process and state of nonlinear dynamic model and the process noise $\mathbf{w}_k$. The process noise $\mathbf{w}_k$ consists of accelerometer and gyro noise $\mathbf{w}_{a,k}^b$ and $\mathbf{w}_{g,k}^b$ respectively identified in Equation (3.10).

$$\begin{aligned} \mathbf{w}_{a,k}^b &\sim N(0, \mathbf{Q}_a), \qquad \mathbf{Q}_a = \sigma_a^2 \mathbf{I}_3 \\ \mathbf{w}_{g,k}^b &\sim N(0, \mathbf{Q}_g), \qquad \mathbf{Q}_g = \sigma_g^2 \mathbf{I}_3. \end{aligned} \tag{3.10}$$

The process noise $\mathbf{w}$ is modelled as zero mean Gaussian noise with standard deviation strength σ, that is:

$$\mathbf{E}[\mathbf{w}_k] = 0$$
$$\mathbf{E}[\mathbf{w}_k \mathbf{w}_k^T] = \mathbf{Q}_k = \begin{bmatrix} \mathbf{Q}_a & 0 \\ 0 & \mathbf{Q}_g \end{bmatrix}. \tag{3.11}$$

Using the position, velocity and orientation navigation equations, the non-linear vehicle process/prediction model according to the Earth-Fixed Local-Tangent frame becomes:

$$\underbrace{\begin{bmatrix} \mathbf{p}_k^n \\ \mathbf{v}_k^n \\ \boldsymbol{\psi}_k^n \end{bmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{bmatrix} \mathbf{p}_{k-1}^n + \mathbf{v}_{k-1}^n \Delta t \\ \mathbf{v}_{k-1}^n + (\mathbf{C}_{b,k-1}^n \mathbf{f}_k^b + \mathbf{g}^n)\Delta t \\ \boldsymbol{\psi}_{k-1}^n + (\mathbf{E}_{b,k-1}^n \boldsymbol{\omega}_k^b)\Delta t \end{bmatrix}}_{\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k)} + \underbrace{\underbrace{\begin{bmatrix} 0_{3,3} & 0_{3,3} \\ \mathbf{C}_{b,k-1}^n & 0_{3,3} \\ 0_{3,3} & \mathbf{E}_{b,k-1}^n \end{bmatrix}}_{\mathbf{G}_{k-1}} \underbrace{\begin{bmatrix} \mathbf{w}_{a,k}^b \\ \mathbf{w}_{g,k}^b \end{bmatrix}}_{\mathbf{w}_k}}_{\mathbf{g}(\mathbf{w}_k)}, \tag{3.12}$$

where $\mathbf{p}_k^n$ and $\mathbf{v}_k^n$ are the position and velocity in the navigation frame and $\boldsymbol{\psi}_k^n$ is the Euler angle, $\mathbf{f}_k^b$ and $\boldsymbol{\omega}_k^b$ are the acceleration and angular-rate measurement in the body frame (IMU output), $\mathbf{C}_b^n$ is the DCM defined in Equation (3.1) and $(\mathbf{E}_n^b)$ is the body rate to Euler rate transformation matrix defined in Equation (3.3). Also $\mathbf{G}_{k-1}$ is the process noise coupling matrix for the linear dynamic system, that is:

$$\mathbf{G}_{k-1} \approx \left. \frac{\partial g(x, w)}{\partial x} \right|_{x = x_{k-1}}. \tag{3.13}$$

The IMU sensors bias will be taken into account and estimated in Section 5.5 of this thesis.

3.4.3 Observation Model

The observation model is linear form as the visual sensor (monocular camera) delivers the direct pose measurements. The observation noise $\mathbf{v}_k$ consists of observation position and orientation noise $\mathbf{v}_{a,k}^b$ and $\mathbf{v}_{\psi,k}^b$ respectively. That is:

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \tag{3.14}$$

$$\begin{aligned} \mathbf{v}_{p,k}^{b} &\sim N(0, \mathbf{R}_p), \qquad \mathbf{R}_p = \sigma_p^2 \mathbf{I}_3 \\ \mathbf{v}_{\psi,k}^{b} &\sim N(0, \mathbf{R}_\psi), \qquad \mathbf{R}_\psi = \sigma_\psi^2 \mathbf{I}_3 \end{aligned} \tag{3.15}$$

where

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & \mathbf{I}_3 \end{bmatrix}. \tag{3.16}$$

Similarly, the observation noise $\mathbf{v_k}$ is modelled as zero mean Gaussian noise with standard deviation strength σ, that is:

$$\begin{aligned} \mathbf{E}[\mathbf{v}_k] &= 0, \\ \mathbf{E}[\mathbf{v}_k \mathbf{v}_k^T] &= \mathbf{R}_k = \begin{bmatrix} \mathbf{R}_p & 0 \\ 0 & \mathbf{R}_\psi \end{bmatrix}. \end{aligned} \tag{3.17}$$

3.5 Summary

This chapter provided an overview of the problem considered in this thesis, the inertial navigation equations and the environment setup in the experiment. The inertial navigation equations were formulated based on fixed navigation frame. The attitudes of the platform were expressed in Euler angle form. By transforming between the body frame and navigation frame, the position and velocity equations were derived. As a result, using the inertial navigation equations, the nonlinear process model was constructed. Finally, the experimental setup provides the direct pose measurements and hence the observation model is in linear form.

Chapter 4

Iterative Smoothing and Outlier Detection

4.1 Overview

This chapter presents the novel iterative smoothing and outlier detection approach using the EKF. This includes the utilisation of the the Biwas-Mahalnabis fixed lag smoother (BMFLS) to estimate the smoothed vehicle state at a fixed-lag time. The Chi-square test is applied to classify measurements through iterations to detect and reject outliers to enhance the navigation accuracy. This chapter includes the BMFLS propagation, algorithm summary, simulation study, experimental results and a discussion where the advantages and limitations of this method are discussed.

4.2 Biswas-Mahalanabis Fixed-lag Smoother (BMFLS)

In this section we present the formulation and details of the BMFLS for nonlinear process model for the problem considered in this thesis. The BMFLS is propagated similarly to the EKF, except with augmented state vector with past predictions $\mathbf{x}_{k[s]} =$

$\{\mathbf{x}_k, \mathbf{x}_{k-1}, \cdots, \mathbf{x}_{k-l}\}$ using $l+1$ lagged states over a fixed size interval (Figure 4.1):

$$\mathbf{\Delta t_{lag}} = \mathbf{l\Delta t}. \tag{4.1}$$

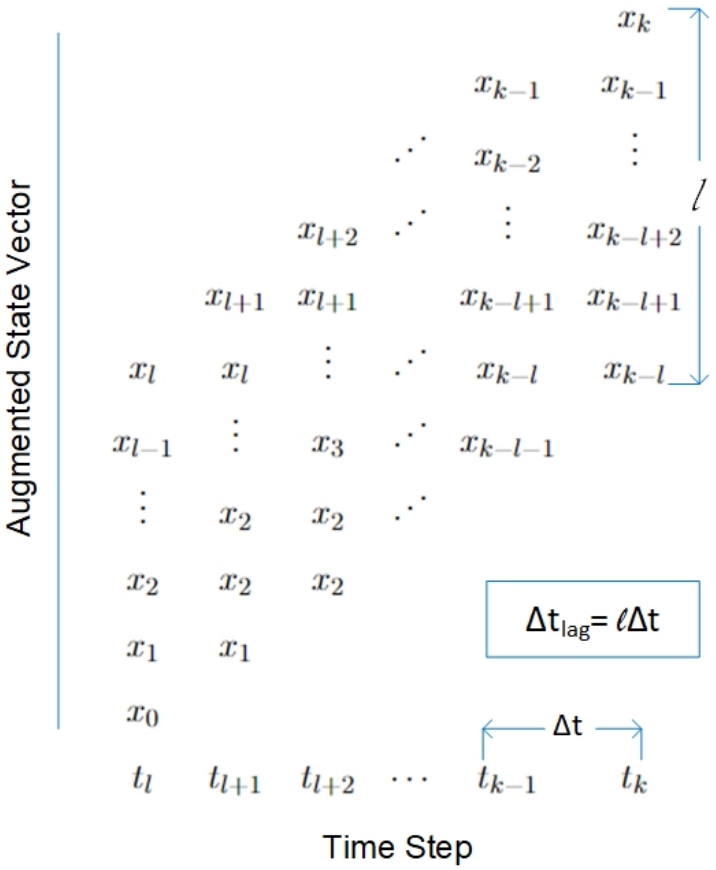

FIGURE 4.1: The BMFLS augmented state vector.

Figure 4.2 represents a summary cycle of the BMFLS where after the prediction/update cycle, states are augmented and the smoothed state of interest is stored. We denote the prior smoothed state vector as:

$$\hat{\mathbf{x}}_{k(-),s} = \begin{bmatrix} \hat{\mathbf{x}}_{k,EKF} \\ \hat{\mathbf{x}}_{k-1,EKF} \\ \vdots \\ \hat{\mathbf{x}}_{k-l,EKF} \end{bmatrix} \tag{4.2}$$

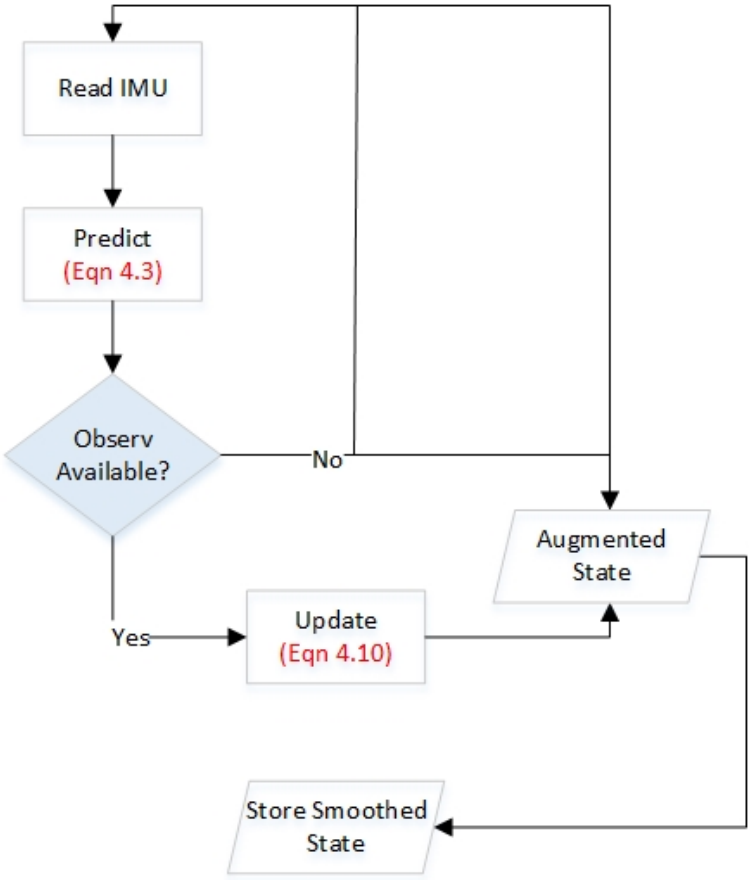

FIGURE 4.2: A representation of the BMFLS cycle

where the current state $\hat{\mathbf{x}}_{k(-)}$ is estimated using the nonlinear dynamic model:

$$\hat{\mathbf{x}}_{k(-)} = \mathbf{f}(\hat{\mathbf{x}}_{k-1(+)}, \mathbf{u}_k) \tag{4.3}$$

$$\hat{\mathbf{P}}_{k(-)} = \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1(+)}\mathbf{F}_{k-1}^T + \mathbf{G}_{k-1}\mathbf{Q}_k\mathbf{G}_{k-1}^T\boldsymbol{\Delta}\mathbf{t}, \tag{4.4}$$

where $\mathbf{F}_{k-1}$ is the Jacobian of the process transition function which is constructed as:

$$\mathbf{F}_{k-1} = \frac{\partial \mathbf{f}(x, u_k)}{\partial x}\Big|_{x_{k-1}}. \tag{4.5}$$

The dimension of the observation matrix $\mathbf{H}_{m\times n}$ is required where m is the number of measurement variables and n is the number of state variables. Also we need to compute the cutoff block index for state augmentation:

The propagated smoothed covariance matrix is:

$$\hat{\mathbf{P}}_{k[s](+)} = \left[\begin{array}{c|c} \hat{\mathbf{P}}_{k(-)} & \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1[s](+)}(1:n, 1:n\times l) \\ \hline \hat{\mathbf{P}}^{T}_{k-1[s](+)}(1:n, 1:n\times l)\mathbf{F}^{T}_{k-1} & \hat{\mathbf{P}}_{[s]k-1(+)}(1:n\times l, 1:n\times l) \end{array}\right]. \quad (4.6)$$

When the vision measurements are available, the innovation (or error) and its covariance are computed:

$$\mathbf{e}_k = (\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k(-)}) \quad (4.7)$$

$$\mathbf{S}_k = \mathbf{H}\mathbf{P}_{k(-)}\mathbf{H}^T + \mathbf{R_k}. \quad (4.8)$$

The BMFLS state and covariance are then updated using the smoothed Kalman gain,

$$\mathbf{K}_{k[s]} = \hat{\mathbf{P}}_{k[s](-)}(:, 1:n)\mathbf{H}^T\mathbf{S}_k^{-1} \quad (4.9)$$

$$\hat{\mathbf{x}}_{k[s](+)} = \hat{\mathbf{x}}_{k[s](-)} + \mathbf{K}_{k[s]}\mathbf{e}_k \quad (4.10)$$

$$\mathbf{P}_{k[s](+)} = \mathbf{P}_{k[s](-)} - \mathbf{K}_{k[s]}\mathbf{H}\mathbf{P}_{k[s](-)}. \quad (4.11)$$

4.3 Robust-BMFLS Algorithm

This section presents how through iterative smoothing and measurement classification process, outliers are detected and navigation trajectory is enhanced. Figure 4.3 manifests a summary cycle of the robust-BMFLS which is an add-on to the BMFLS cycle from Figure 4.2. A Pseudo-code of the proposed method is outlined in Algorithm 1. In order to achieve a stable solution, we assume that there are some inliers captured in time intervals with large number of outliers.

4.3.1 Algorithm Description

The Algorithm 1 works by initially treating all the raw observation measurements as inliers in the first iteration (line 2). Then, the smoothed states are obtained from the BMFLS function (lines 6-9) as shown in the flow chart in Figure 4.2.

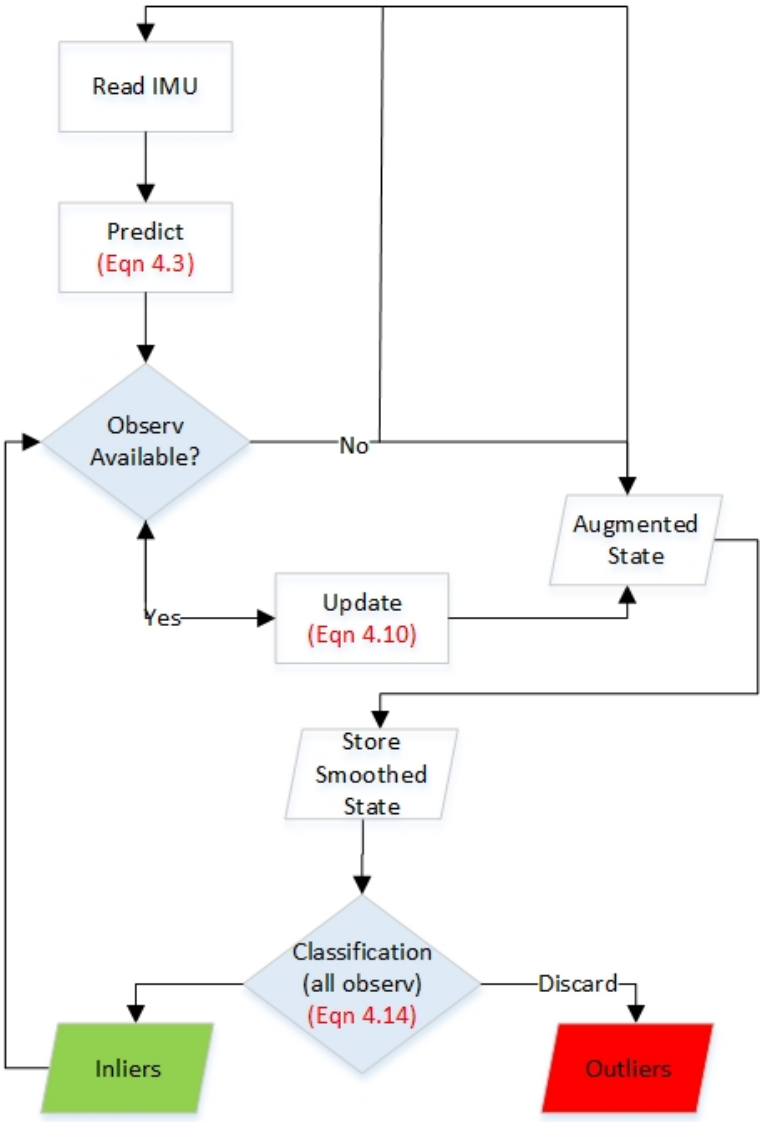

FIGURE 4.3: The iterative smoothing and outlier detection cycle.

Next (still first iteration), inliers and outliers for the entire observations are sorted by computing the Mahalanobis distance (MD) and running through the Chi-square test (lines 12-26). The MD is calculated for each observation using the smoothed output (from current iteration) to perform measurement classification. That is,

$$\mathbf{e}_k = (\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k(-)}) \tag{4.12}$$

$$\mathbf{d}_k = \mathbf{e}_k^T \mathbf{S}_k \mathbf{e}_k \tag{4.13}$$

$$\mathbf{z_k} = \begin{cases} \textit{inlier}, \text{ if } \mathbf{d}_k < \chi_T^2 \\ \textit{outlier}, \text{ if } \mathbf{d}_k \geq \chi_T^2 \end{cases} \tag{4.14}$$

where χ_T^2 is the threshold value selected from Chi-square distribution table based on the degree-of-freedom (DoF) of the observation model (see Section 2.4.1). If the MD is below the threshold, the observation is considered an inlier, otherwise it's declared an outlier and will not be considered in the next iteration.

Also, to make the programming easier, the index of inliers are stored during iteration where $\mathbf{c} = 1$ (*inlier*) and $\mathbf{c} = 0$ (*outlier*) as shown in Equation (4.15). where $\mathbf{c} \in \{0, 1\}$ is the outlier association variable. The variable $\mathbf{c}$ is passed into the **BMFLS** function in next iteration (line 8). Note, initially the values of c are all ones because all measurements are treated as inliers as stated above.

$$\mathbf{inlier\_index(k)} = \begin{cases} 1, \text{ if } \mathbf{d}_k < \chi_T^2 \\ 0, \text{ if } \mathbf{d}_k \geq \chi_T^2. \end{cases} \tag{4.15}$$

In the second iteration, the trajectory is refined without outliers captured in the previous step. Using smoothed states in the second iteration, all the visual measurements are reclassified to separate inliers and outliers. In the next iteration, the smoothed state are predicted using the inliers captured in the previous iteration. This smoothing and observation re-classification process are repeated until the inliers converge where all potential outliers are screened out.

4.4 Simulation

A simulation data corresponding to the dynamic and observation model defined in Section 3.4.2 and 3.4.3 is generated to examine this work as compared to the ground-truth. The simulation study is carried out for two reasons: 1) obtaining ground truth in the experiment was not feasible, 2) to further validate the work. Figure 4.4 illustrates the 3D position ground-truth trajectory with raw measurements generated for the simulation study with outliers injected at certain intervals.

Algorithm 1: Iterative Smoothing and Outlier Detection

Initialisation: x_0, P_0, Q, R, $\chi^2_{threshold}$;
inlier_index = ones(1, len);
for *iter=1:N* **do**
    /*Obtaining smoothed state without outliers*/;
    $x_s = []$;
    for $i = 1 : len$ **do**
        c = inlier_index(i);
        $[x_s]$ = BMFLS_Function(...., c);
    end
    /*Reclassification of all raw observations*/;
    inlier = []; outlier = [];
    for $k = 1 : len$ **do**
        if *measurement available* **then**
            $e_{k[s]} = (z_k - H\hat{x}_{k[s]})$;
            $S_{s[k]} = HP_{k[s]}H^T + R$;
            $d_k = e^T_{k[s]} S_{k[s]} e_{k[s]}$;
            /*perform outlier gating test*/;
            if $d_k > \chi^2_{threshold}$ **then**
                inlier_index(k) = 0;
                outlier(:, k) = z_k;
            else
                inlier_index(k) = 1;
                inlier(:, k) = z_k
            end
        end
    end
    plot(states);
end

4.4.1 Results

This section presents the results of the robust-BMFLS using the simulation dataset. Figure 4.5 illustrates position and orientation curves during first and fifth iteration for a fixed-lag size of 60 steps. Also, the root-mean square error (RMSE) at each iteration is summarised in Table 4.1.

As shown in iteration 1, during about $800 - 1000$ steps, some measurements are wrongly classified. This has resulted the orientation curve to drift away from ground-truth and a large rotational RMSE (0.6462 *rad*). However, in the fifth iteration, the measurements in this interval $800 - 1000$ steps have been correctly reclassified which has reduced the

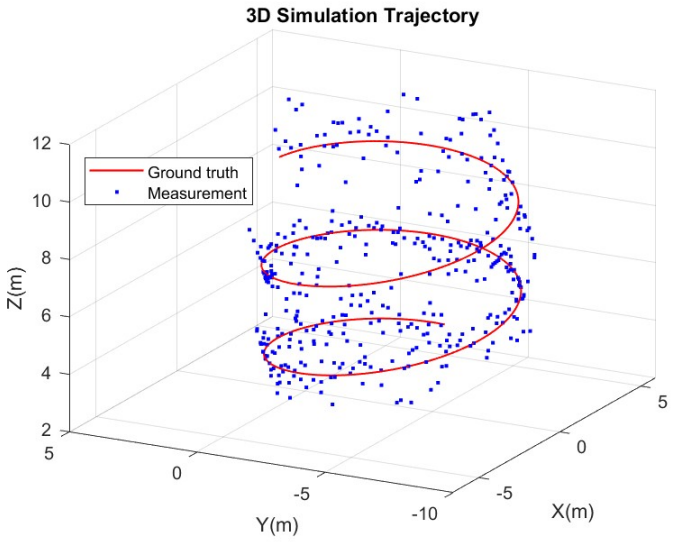

FIGURE 4.4: 3D trajectory of the simulated data with ground-truth.

RMSE (0.1406 *rad*) where the potential outliers have been captured and the solution has converged. The simulation dataset contains a moderate level of outliers which is handled well through iterative smoothing with Chi-square test.

In addition, the 2D projected position trajectory after outlier rejection is shown with comparison to EKF in Figure 4.6. Further, the RMSE for different lag sizes are manifested in Table 4.2. The EKF has the largest RMSE, while lag size of 0 is simply the EKF which after iterative outlier removal has improved the RMSE. Furthermore, the lag size of 100 has the lowest RMSE. This reinforces the usefulness of a large window and in particular smoothing in enhancing the navigation trajectory, however with an increase in computation.

TABLE 4.1: The RMSE at each iteration for fixed-lag size of 60 steps.

Iteration Number	RMSE(m)	RMSE(rad)
1	0.4312	0.6462
2	0.6953	0.1405
3	0.5152	0.1406
4	0.4405	0.1406
5	0.4291	0.1406

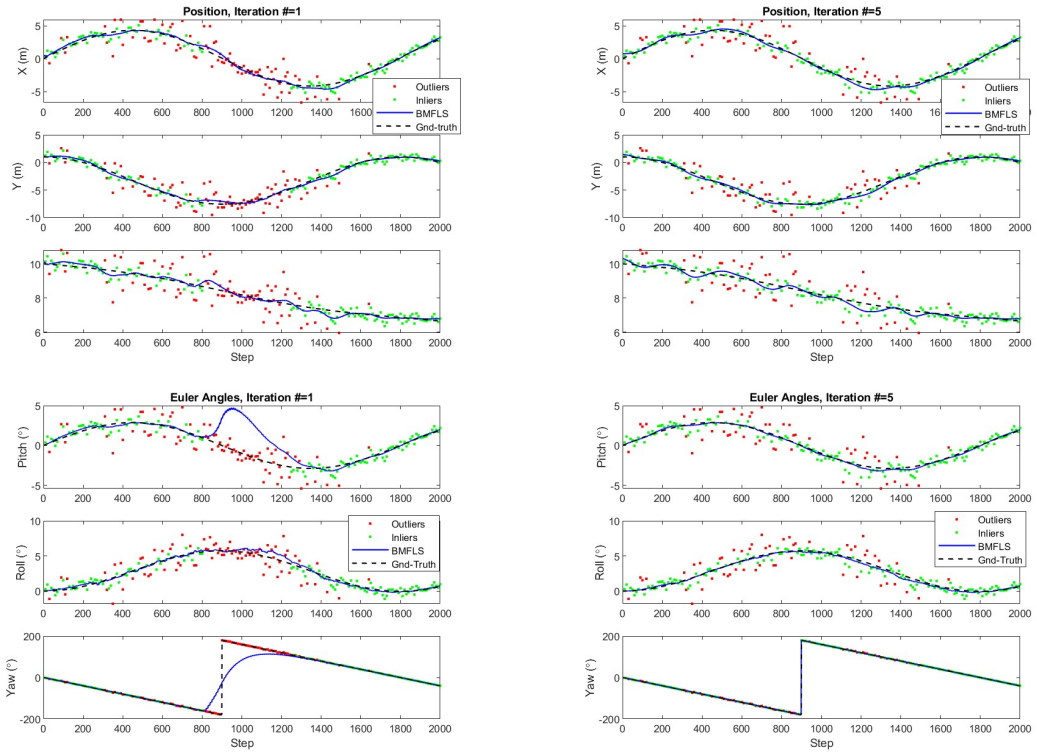

FIGURE 4.5: The first and fifth iteration results using simulation dataset for the position and Euler angles.

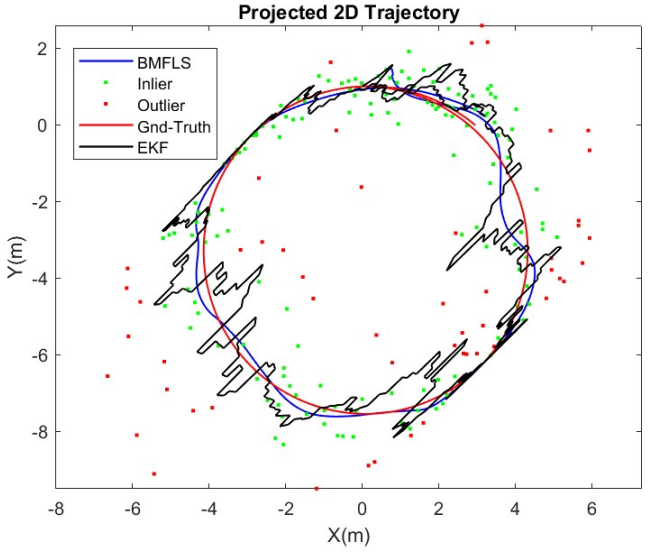

FIGURE 4.6: 2D projected position trajectory after outlier rejection.

TABLE 4.2: The RMSE comparison for different fixed-lag sizes after outlier rejection(simulation).

Lag Size	RMSE(m)	RMSE(rad)
EKF	9.7803	1.7724
0	0.8696	1.0172
20	0.6644	0.1708
80	0.5054	0.2012
100	0.3313	0.1405

4.5 Particle Filter

This section presents the SIR PF results using the simulation dataset. The SIR filter pseudocode is outlined in algorithm 2. It works by selecting the number of particles N_p. During each cycle (lines 5-11), each particle state is predicted and the associated weight is calculated using the likelihood function. Next, all weights are normalised (lines 13-15) and the effective sample size $\hat{N}_{eff}$ is computed. If the $\hat{N}_{eff}$ falls below a threshold, then resampling is carried out (lines 17-19). Finally, the state at time step k is estimated (line 20).

Figure 4.7 illustrates the SIR filter position trajectory (blue line). As shown, the SIR filter degrades in presence of outliers where it eventually started to diverge. Similarly, the SIR filter is also claimed to diverge rapidly in [32] with small level of outliers. Also, according to [37], the PF performs well 3D state space, but becomes less useful in six-dimensional state. This demonstration manifests the vulnerabilities of the SIR filter to outliers.

Algorithm 2: SIR filter pseudocode

$x_{pp} = x_0 * ones(1, N_p)$;
$N = \#$ IMU measurement;
$w_1^i = ones(1, N_p)/N_p$;
for $k = 1 : N$ **do**
  for $i = 1 : N_p$ **do**
    $x_k^i = f(x_{k-1}^i) + randn(\tilde{w})$;
    if *Measurement available* **then**
      Calculate: $w_k^i = p(z_k|x_k^i)$
    end
    $x_{pp}(:, i) = x_k^i$;
  end
  $t = SUM[w_k^i]_{i=1}^{N_p}$;
  for $i = 1 : N_p$ **do**
    Normalise weights: $w_k^i = t^{-1} w_k^i$;
  end
  Calculate: $\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{Np} (w_k^i)^2}$;
  if $\hat{N}_{eff} < N_{thr}$ **then**
    $\{x_k^i, w_k^i\}_{i=1}^{N_p} = resample(x_{pp}, w_k, N_p)$;
  end
  State est: $\hat{x}_k = \sum_{i=1}^{Np} (w_k^i x_k^i)$;
end

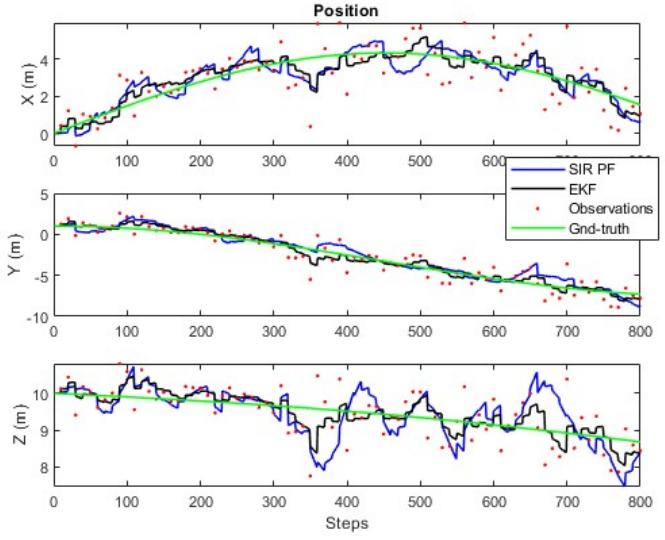

FIGURE 4.7: The SIR filter position trajectory using simulation dataset. Due to presence of outliers, the solution is diverging.

4.6 Experimental Results

This section presents the results achieved by robust-BMFLS using the experimental data. The details of the experimental setup is outlined in Section 3.3. The robot was manually controlled to maintain the hovering position under a current disturbance. Due to difficulties, it was not feasible to obtain the ground-truth trajectory. The experiment lasted about 50 minutes. The IMU and monocular camera operate at 252 Hz and 26 Hz respectively. The iterative smoothing and outlier rejection results using the experimental dataset

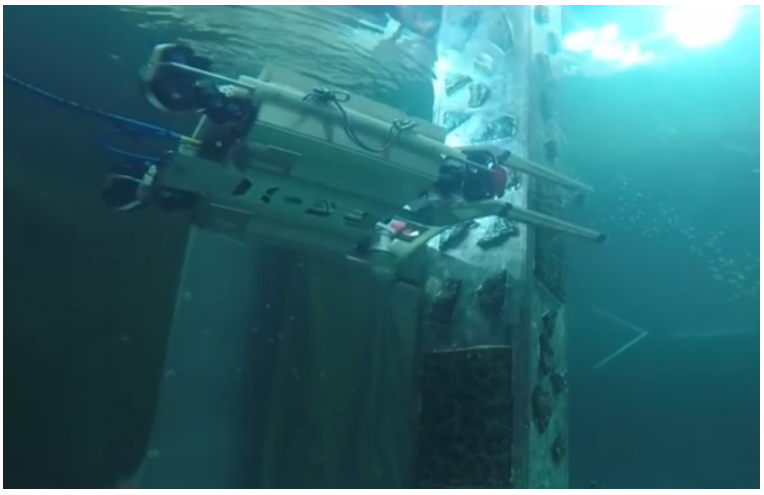

FIGURE 4.8: The SPIR3 performing underwater inspection and maintenance task in a test-water tank facility.

is illustrated in Figure 4.9. The figure shows the position and orientation curves between the first and sixth iteration of the algorithm where the red dots indicate outliers while green dots represent inliers. Also, the ratio of outliers and inliers during each iteration is captured in Table 4.3.

During the first iteration, the trajectory (blue line) is estimated using all the raw vision observations. Using the pose estimations from iteration 1, the measurement observations are separated into outliers and inliers. In the experimental result, there are some heavy intense outlier presence in some intervals such as in the $(0-6)$ seconds. It can be observed there are some measurements that are being wrongly classified as outliers (35%) and vice-versa in the first iteration.

In the second iteration, It is seen, some measurements have been correctly classified such as in the $(0-6)$ seconds interval where the ratio of outliers is 33 %. Moreover, during each iteration, the observations are seen gradually being classifieds correctly due to performing smoothing without outliers captured from previous step. Finally, it's visible in the sixth iteration, the vision observations are being correctly classified and there are 29 % outliers is declared where the smoothed trajectory (blue line) is running through inlier measurements.

The results indicate the proposed approach is capable of handling outliers in an efficient manner without the need for tuning the noise covariance matrices for the process and observation model. On the other hand, the results from first iteration reinforces the limitation of the Chi-square test for outlier rejection without tuning the process and observation model noise covariance matrices.

Furthermore, Figure 4.10 illustrates the proposed solution outlier handling capability even time periods with high outlier density such as in the $(0-6)$ seconds. The high outlier presence in this period is due to turbulence created in water when the robot was switched on. In such instance, the camera is confused between the different ARTag markers resulting in high number of outliers during this period. In addition, a comparison between the ordinary BMFLS and robust-BMFLS are shown in Figures 4.11 and 4.12. It is evident an ordinary (non-robust) BMFLS navigation trajectory degrades with erroneous measurements.

TABLE 4.3: The inlier and outlier ratio at each iteration.

Iteration Number	Inlier %	Oulier %
1	65	35
2	67	33
3	68	32
4	70	30
5	71	29
6	71	29

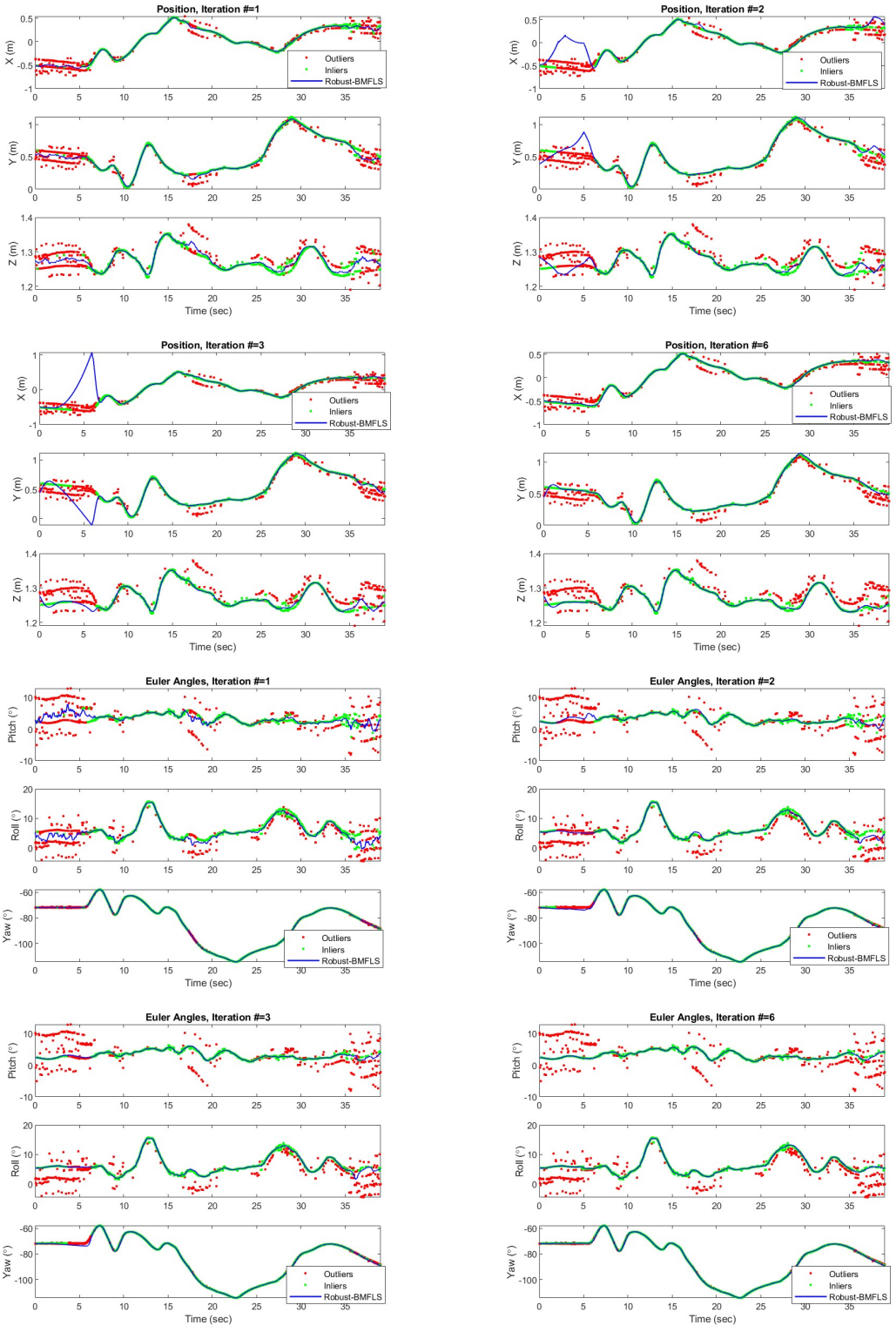

FIGURE 4.9: The first to sixth iteration results of the method for the position and Euler angles. It can be seen that the set of outliers gradually decreases as the iteration progresses, resulting in a better smoothing result at the sixth iteration.

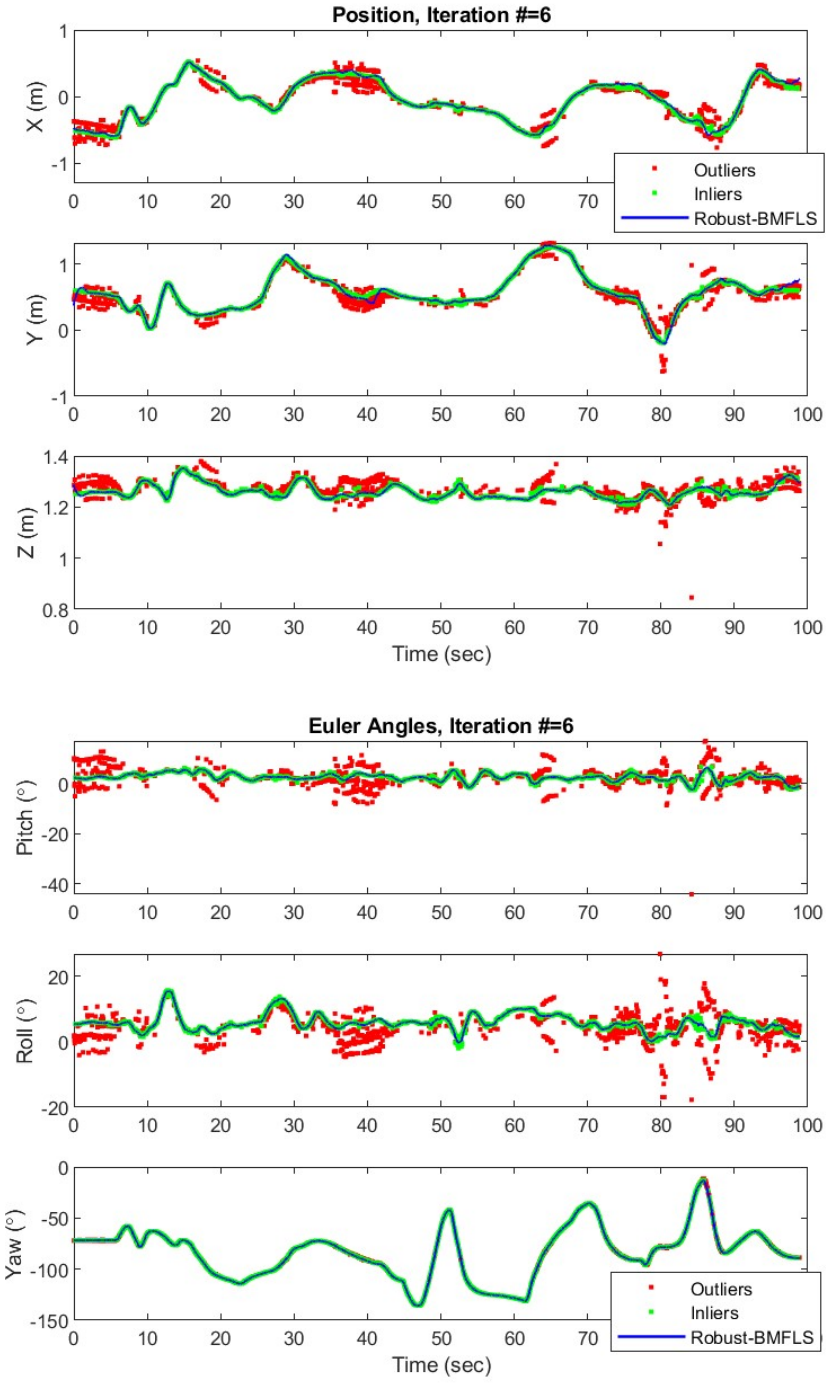

FIGURE 4.10: A better visualisation showing the separation of observations for period of (100s). It can be seen there are time periods with high outliers density.

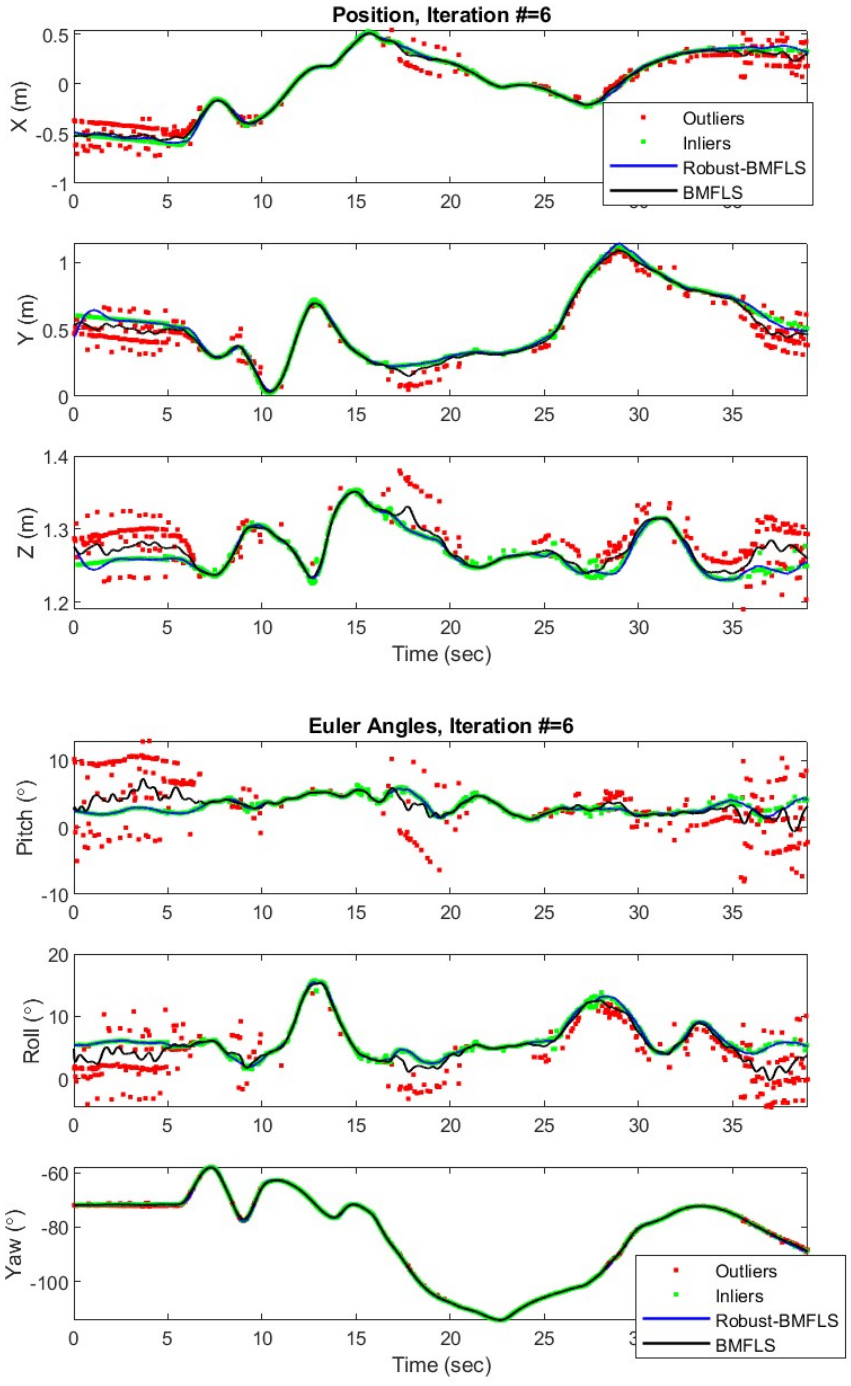

FIGURE 4.11: An illustration of an ordinary (non-robust) BMFLS performance where the trajectory is pulled towards the erroneous measurements.

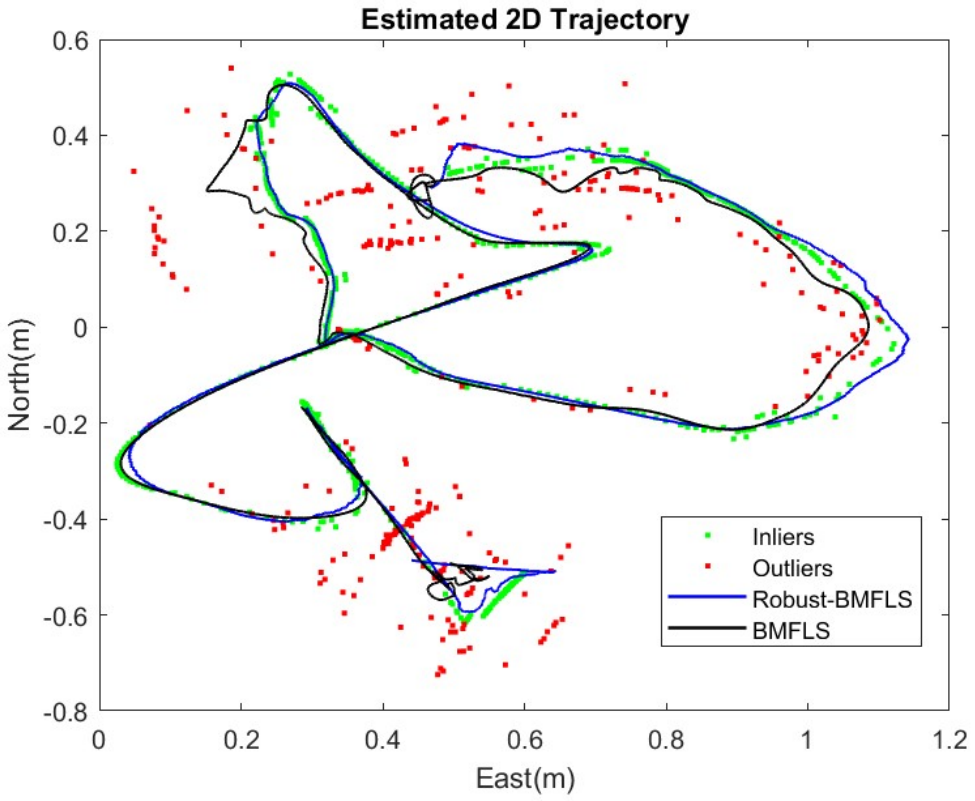

FIGURE 4.12: A 2D projected trajectory showing an ordinary (non-robust) BMFLS being impacted by outlier measurements.

4.7 Discussion

The experimental results in Section 4.6 has shown the raw vision observations contain erroneous measurements in some regions. This is due to the poor visibility and camera confusion between the different markers, causing frequent outliers in the measurements. But, through iterative smoothing and outlier rejection, the impact of wrong measurements on navigation accuracy is mitigated. However, the methodology demonstrated in this chapter applies the Chi-square test to detect and reject outliers at each iteration makes this a simple algorithm to implement.

On the other hand, It was pointed out the visual observations were wrongly classified in some instances. This reinforces the limitation of Chi-square test where it does not distinguish measurement observation correctly when there are too many outlier present

in a single iteration. Thus, through iterations, measurements are gradually reclassified where potential outliers are screened out. This reinforces one limitation of the iterative robust-BMFLS where it may not detect potential outliers in single iteration and requires few more iterations.

While the proposed method works well when there are moderate or even large number of outliers, if there are few or no inliers detected in a time periods with high outlier density, it may lead to divergence until the trajectory is corrected again. This may be due to high cluster of outliers affect the mean and variance of the dataset. This is resolved by increasing the threshold in such time intervals for more inliers to appear thus reducing the influence of outliers to recover the trajectory.

Finally, the proposed methodology solution is simple to implement. The augmented state is run similar to EKF, yielding a computationally efficient performance. A large lag size does provide a better smoothed trajectory, but just means more computation. The efficiency of the robust-BMFLS is discussed in Chapter 5.

4.8 Summary

This chapter has presented the novel iterative smoothing and outlier detection by utilising the BMFLS and using the Mahalanobis gating test to capture outliers through iterations with classification. Both simulation and experimental dataset were used to validate the proposed method. We have seen the methodology is simple that works well in handling moderate to large level of outliers. The limitation is more iterations are required in time intervals with large outlier ratio for the solution to converge.

Chapter 5

Incremental Robust Solution with Expectation-Maximisation

5.1 Overview

This chapter presents an incremental robust solution with EM using the nonlinear least-squares optimisation or the maximum a posterior (MAP) approach. A robust approach was introduced in [30] to detect false loop closures in the pose-graph SLAM problem. We extended this concept navigation tracking problem and introduce a RSWF approach for fixed-lag time applications and to ease the computation burden in full-batch estimation method. The effect of different window sizes and the update periods are studied to examine accuracy and efficiency achieved.

5.2 Robust Localisation with EM

The robust optimisation approach is represented as the Bayesian network shown in Figure 5.1, where the $X = [x_1, x_2, x_3, ..., x_n]$ are the robot poses and $Z = [z_1, z_2, z_3, ..., z_n]$ are the measurement observations. We assign weights to each observation $W = [W_1, W_2, W_3, ..., W_n]$ where $W \in [0, 1]$. The value of W determines the weight of the observation in the optimisation.

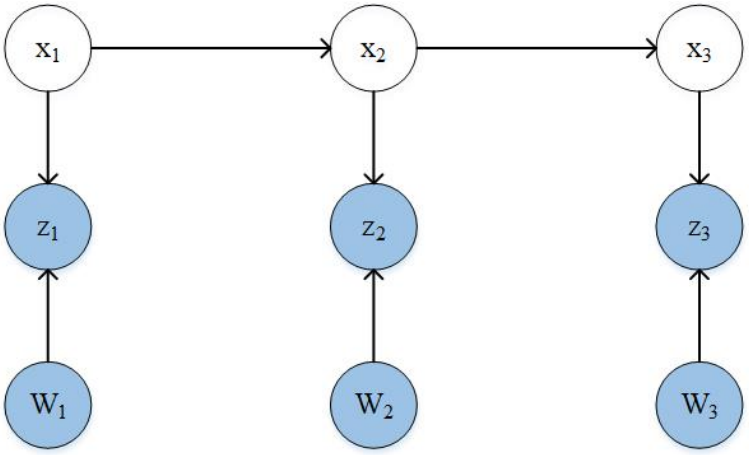

FIGURE 5.1: Bayesian network representing the navigation problem. W_k is the assigned weight for each measurement z_k in the optimisation.

The optimisation framework requires finding the MAP, that is:

$$\hat{x} = \arg\max_{x} p(\mathbf{x}|\mathbf{w}, \mathbf{z}). \tag{5.1}$$

The MAP in Equation (5.1) is solved via expectation-maximisation outlined in Section 5.2.1 and 5.2.2.

5.2.1 Expectation Step

In the expectation step, the weight variable W is computed by maximising $p(W|x, z)$.

$$\begin{aligned} W &= \arg\max_{x} p(\mathbf{w}|\mathbf{x}, \mathbf{z}) \\ &= \frac{p(\mathbf{w}, \mathbf{x}, \mathbf{z})}{p(\mathbf{x}, \mathbf{z})} \\ &\propto p(\mathbf{z}|\mathbf{w}, \mathbf{x}). \end{aligned} \tag{5.2}$$

From the expectation maximisation formulation in [30], the weight W_k is found to be the Cauchy weight function:

$$W_k = \frac{C^2}{C^2 + ||z_k - f(x_k)||^2_{R_k}} \tag{5.3}$$

where C is a constant and $||z_k - f(x_k)||^2_{R_k}$ is the Mahalanobis distance. It can be observed the Cauchy weight function varies between 1 and 0. With increasing error (Mahalanobis distance dominates) and the weight function reaches close to zero.

The weights assigned to the observations are iteratively computed from the errors between the predicted and observed vehicle poses, until the weights of all observations converge.

5.2.2 Maximisation Step

In this step, the MAP is computed using the weight value W from expectation step described in Section 5.2.1. Using Bayes' rule, MAP is obtained as follows:

$$\begin{aligned} \hat{x} &= \arg\max_x p(\mathbf{x}|\mathbf{w}, \mathbf{z}) \\ &\propto p(\mathbf{z}|\mathbf{w}, \mathbf{x})p(\mathbf{x}) \\ &\propto \prod_{k=0}^{K} p(\mathbf{z_k}|\mathbf{w_k}, \mathbf{x_k}) \prod_{k=0}^{K} p(\mathbf{x_k}|\mathbf{x_{k-1}}). \end{aligned} \tag{5.4}$$

5.3 Batch Estimation Optimisation

This section details the weighted NLS full batch estimation theoretical work used for the approach in this chapter. The solution requires an initial estimate of the states which are obtained using the EKF which are then smoothed using the weighted NLS.

5.3.1 Initialisation

The nonlinear least-squares requires an initial estimate of the states $\mathbf{x}^0_{0:K}$ which are obtained from using EKF. Every time a measurement is available, the position and orientation states are corrected.

5.3.2 Nonlinear Least-Squares Optimisation

The objective function that requires to be minimised is defined in Equation (5.5) which is the sum of squares of the prediction and observation models error. That is:

$$[\delta x_k^*] = \arg\min_{\delta x_k} \sum_{k=0}^{K} (L_{v,k}(x) + L_{z,k}(x)) \tag{5.5}$$

where $L_{v,k}(x)$ is the prediction model squared error and $L_{z,k}(x)$ is the weighted observation model squared error as stated below:

$$L_{v,k}(x) = \frac{1}{2} e_{v,k}(x)^T Q_{v,k}^{-1} e_{v,k}(x) \tag{5.6a}$$

$$L_{z,k}(x) = \frac{1}{2} e_{z,k}(x)^T W_k R_{z,k}^{-1} e_{z,k}(x). \tag{5.6b}$$

The linearised motion and observation model error at time k around the operating point x_{op} are as follows, respectively:

$$e_{v,k}(x_{op} + \delta x) \approx \begin{cases} e_{v,0}(x_{op}) + \delta x_0, & k = 0. \\ e_{v,k}(x_{op}) + F_{k-1}\delta x_{k-1} - \delta x_k, & k = 1, ..., K. \end{cases} \tag{5.7a}$$

$$e_{z,k}(x_{op} + \delta x) \approx e_{z,k}(x_{op}) - H_k \delta x_k, \ k = 0, ..., K \tag{5.7b}$$

where

$$e_{v,k} = \begin{cases} \hat{x}_0 - x_0, & k = 0. \\ f(x_{k-1}, u_k) - x_k, & k = 1, ..., K. \end{cases} \tag{5.8a}$$

$$e_{z,k} = z_k - h(x_k), \ k = 0, ..., K \tag{5.8b}$$

where x_{op} denotes the operating point. We solve for δx at each iteration until the algorithm converges.

$$\delta x = \begin{bmatrix} \delta x_0 & \delta x_1 & \delta x_2 & \dots & \delta x_K \end{bmatrix}^T. \tag{5.9}$$

The combined prediction and observation error is $e(x_{op})$:

$$e(x_{op}) = \left[\begin{array}{cccc|cccc} \delta_{v,0}(x_{op}) & \delta_{v,1}(x_{op}) & \dots & e_{v,K}(x_{op}) & e_{z,0}(x_{op}) & e_{z,1}(x_{op}) & \dots & e_{z,K}(x_{op}) \end{array} \right]^T. \tag{5.10}$$

The stacked version of the problem can be written and solved iteratively. The structure of the combined motion and observation model Jacobians J matrix can be represented as:

$$J = \left[\begin{array}{ccccccc} I & & & & & & \\ -F_0 & I & & & & & \\ & -F_1 & \ddots & & & & \\ & & \ddots & & & & \\ & & & & I & & \\ & & & & -F_{K-1} & I & \\ \hline H_0 & & & & & & \\ & H_1 & & & & & \\ & & H_2 & & & & \\ & & & & \ddots & & \\ & & & & & & H_K \end{array} \right] = \left[\begin{array}{c} J_v \\ \hline J_z \end{array} \right] \tag{5.11}$$

$$\Lambda = diag(P_0^{-1}, \tilde{Q}_1^{-1}, \cdots, \tilde{Q}_K^{-1}, W_0 R_0^{-1}, \cdots, W_K R_K^{-1}) \tag{5.12}$$

$$\underbrace{(J^T \Lambda J)}_{\text{information matrix}} \quad \delta x^* = \underbrace{J^T \Lambda e(x_{op})}_{\text{information vector}} \tag{5.13}$$

$$x^s = x_{op} + \delta x^*. \tag{5.14}$$

Notice the covariance matrix of the process noise $\tilde{Q} = G_{k-1} Q_k G_{k-1}^T$ is singular and not invertible. We can add a small diagonal matrix ΔI, to allow the covariance matrix become invertible.

5.4 Sliding Window Filter

As stated in literature review, in order for bath-estimation to work efficiently over long period, the state vector cannot grow without bound. The SWF approach is one novel solution that can resolve this. This section presents the Schur complement in the SWF to perform marginalisation to discard oldest poses and maintain prior information. The robust SWF in this work iterates over a fixed-size window to perform smoothing and measurement classification.

5.4.1 Marginalisation

Marginalising out parameters is equivalent to applying the Schur complement to the linear equation. For example, given the system:

$$\underbrace{\begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \delta x_1^* \\ \delta x_2^* \end{bmatrix}}_{\delta x^*} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}}_{b} \tag{5.15}$$

reducing the parameter x_1 (marginalised state) into x_2 gives:

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} - A_{12}^T A_{11}^{-1} A_{12} \end{bmatrix} \begin{bmatrix} \delta x_1^* \\ \delta x_2^* \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 - A_{12}^T A_{11}^{-1} b_1 \end{bmatrix} \tag{5.16}$$

where the matrices A and b in Equation (5.15) represent the terms $(J^T \Lambda J)$ and $(J^T \Lambda e(x_k))$ in Equation (5.13) respectively. Finally, we solve for δx_2^*.

5.4.2 RSWF Algorithm

Here, we provide a summary of the SWF algorithm"

- Add the new pose parameters: after completing $k-1$ steps, we apply the process model using $\mathbf{x_k} = \mathbf{f}(\mathbf{x_{k-1}}, \mathbf{u_k})$ and update the information matrix (underlined in Equation (5.13);

- Remove parameters: if there are more than k poses, we marginalise out a defined number[1] of old poses using the Schur complement;
- Update parameters: we now solve for δx_2^* defined in Equation (5.16) using Gauss Newton and apply the outlier detection method.

Algorithm 3 outlines the pseudo code for the RSWF navigation. Lines 20-26 are the expectation step where W_k is calculated based on the smoothed output from previous stem x^s. Lines 30-36 are the maximisation step where the pose predictions are updated. Next, the measurement classification occurs (lines 39-45) by going through the entire measurement observations in the current window and recalculating the weight W_k based on the current smoothed trajectory. In here, the measurements with weights less than the threshold are removed. The EM process is repeated until the weights converge (line 8). The EM process helps in preventing local minima. Finally, the current window is then slided forward (line 7) with the defined number of oldest poses discarded.

5.4.3 Update Rate

As stated in Chapter 2 literature, researchers have considered compressed-SLAM or incremental smoothing [16] [17] to reduce the computational complexity of SLAM. We apply a similar concept in this incremental robust approach. Because the states in the window of optimisation remain almost unchanged, the window of optimisation could be slided forward at more than one step.

The number of steps we slide the window forward sets the update rate or marginalisation size. The maximum number number of steps that the RSWF with window size ws can moved up is half of optimisation window size ($ws/2$).

[1]This is the update rate, e.g update every 10 steps.

Algorithm 3: Incremental Robust Navigation Pseudo Code

Window size ws;
Input x_0 : initial state z, all measurements I;
Output x_s : smoothed trajectory;
steps = marginalisation size or update rate;
N = # IMU measurement;
$W' = 0$; $W = I$; $I' = I$;
for $i = 0 : steps : N$ **do**
  while |W − W' | > ν **do**
    $W' = W$;
    /* Classification EM iterations/*;
    while $|\delta| > \eta$ **do**
      /* SWF/*;
      for $k = i : i + ws$ **do**
        predict state: $x_{k(-)} = f(x^s_{k-1})$;
        calculate J_v and;
        Λ_Q = blkdiag(Λ_Q, Q^{-1});
        calculate $e_{v,k}$ = f(x^s_{k-1}) - $\hat{x}_k$;
        set $e_v = [e_v, e_{v,k}]$;
        /*Expectation step*/;
        if *measurement k in* ∈ I' **then**
          compute W_k with Equation (5.3);
          calculate J_z and;
          Λ_R = blkdiag($\Lambda_R, W_k R^{-1}$);
          calculate $e_{z,k} = z_k$ - $H\hat{x}_k$;
          set $e_z = [e_z, e_{z,k}]$;
        end
      end
      /*Maximisation step/*;
      Assemble ;
      J = $[J_v; J_z]$;
      $\Lambda = [\Lambda_Q; \Lambda_R]$;
      $e = [e_v; e_z]$;
      Set up Schur complement (Equation (5.16));
      and solve for δx^*;
      $J^T \Lambda J \ \delta x^* = J^T \Lambda e$;
      calculate $[x^s]^T = [x^s]^T + \delta x^*$;
    end
    /* Remove outliers */;
    for *all measurements in* I **do**
      compute W_k with Equation (5.3);
      if $W_k < \omega$ **then**
        Remove measurement k from I';
      end
      $X = [x^s]^T$, $W = W_k$;
    end
  end
  Store states and remove old steps poses ;
end

5.4.4 Window Size

Additionally, another factor that influence the computational cost is the size of the window of optimisation. Small window size would mean faster computation which we shall investigate the accuracy loss for reduced window sizes.

5.5 Bias Inclusion

In the method described in this chapter, the IMU bias is also included in the state vector. The new state vector $\mathbf{x_v}$ is:

$$\mathbf{x_v} = \begin{bmatrix} \mathbf{p}^n & \mathbf{v}^n & \boldsymbol{\Psi}^n & \mathbf{b}_a^b & \mathbf{b}_g^b \end{bmatrix}^T_{1\times 15} \tag{5.17}$$

where

- Accelerometer bias in the body frame: $\mathbf{b}_a^b = (\mathbf{b}_{ax}, \mathbf{b}_{ay}, \mathbf{b}_{az})$,
- Gyroscope bias in the body frame: $\mathbf{b}_g^b = (\mathbf{b}_{gx}, \mathbf{b}_{gy}, \mathbf{b}_{gz})$.

As a result, the full inertial navigation equations become:

$$\mathbf{p}_{k+1}^n = \mathbf{p}_k^n + \mathbf{v}_k^n \Delta t, \tag{5.18}$$

$$\mathbf{v}_{k+1}^n = \mathbf{v}_k^n + (\mathbf{C}_b^n[\mathbf{f}_k^b - \mathbf{b}_{g,k}^b] + \mathbf{g}^n)\Delta t, \tag{5.19}$$

$$\boldsymbol{\psi}_{k+1}^n = \boldsymbol{\psi}_k^n + \mathbf{E}_b^n[\boldsymbol{\omega}_k^b - \mathbf{b}_{g,k}^b]\Delta t. \tag{5.20}$$

The bias is estimated using the FOGM model. The FOGM stochastic process is suitable for auto-correlated sequences [38]. Bias instability is auto-correlated and hence is modelled using FOGM model.

The additional state errors (bias) are augmented to the main 9-states in the state vector. Therefore the state vector is partitioned and the linearised navigation equation is:

$$\left[\begin{array}{c} \mathbf{x}_{ins} \\ \hdashline \mathbf{x}_b \end{array}\right]_{k+1} = \left[\begin{array}{c:c} \mathbf{F}_{ins} & \mathbf{C} \\ \hdashline 0 & \mathbf{F}_b \end{array}\right]_k \left[\begin{array}{c} \mathbf{x}_{ins} \\ \hdashline \mathbf{x}_b \end{array}\right]_k + \left[\begin{array}{c:c} \mathbf{G}_{ins} & 0 \\ \hdashline 0 & \mathbf{I} \end{array}\right]_k \left[\begin{array}{c} \mathbf{w}_{ins} \\ \hdashline \mathbf{w}_b \end{array}\right]_k \tag{5.21}$$

where the state transition parameter $\mathbf{F}_b = \mathbf{e}^{-\Delta t/\tau}$ [39], $\mathbf{\Delta t}$ is the sampling time. Note the subscript ins in Equation (5.21) refers to parameters related to the 9 main body states (position, velocity and orientation).

5.6 Simulation Results

The simulation dataset generated in Chapter 4 is used to verify the proposed method in this chapter as well. A number of different window sizes and update periods are studied with the RMSE computed with reference to the ground-truth.

Figure 5.2 is the projected estimated 2D trajectories of the entire simulation for different window sizes and the full-batch (partial) after outlier rejection with update rate at every 10 steps. By visual inspection, it appears the full-batch estimation remains close to the ground-truth followed by the largest window size (100).

In order to do a comparison between the full-batch estimation and RSWF, a computational efficiency factor is computed. The performance run-time efficiency is calculated with reference full-batch estimation run-time, that is:

$$\eta_{eff} = \frac{time_{full} - time_{RSWF}}{time_{full}}. \tag{5.22}$$

The algorithm is implemented and tested in MATLAB using Intel Core i7 (4 Core) 3 GHz processor. Figure 5.3 displays the computational efficiency curve for different window sizes for a particular update period. It is clear from the plot, a small window size with lowest update rate is the most efficient.

Although, an efficient navigation algorithm is desirable, accuracy is another key element that we are interested to maintain. Table 5.1 also provides the RMSE for different window

sizes and update periods. The full-batch techniques yields the lowest RMSE as expected. But, for it can be seen that performing updates at every 10 steps result in a lower RMSE in general compared to executing updates at every 20 steps.

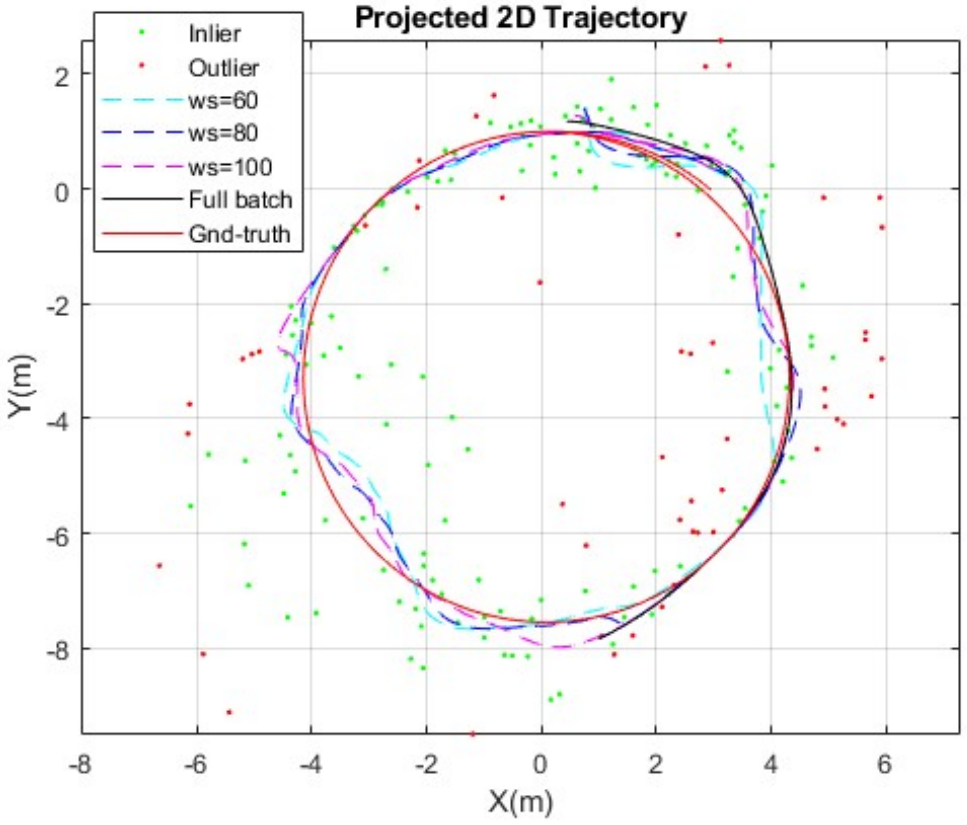

FIGURE 5.2: The estimated trajectory projected in 2D for different window sizes with update period of 10 steps.

TABLE 5.1: The RMSE comparison for different window sizes and update periods after outlier rejection(simulation).

Method	Window size	RMSE(m)	RMSE(Deg)
Full Batch Est	N/A	0.1384	0.0074
SWF(1 step)	60	0.3730	0.3640
	80	0.3159	0.3911
	100	0.3670	0.4542
SWF(10 steps)	60	0.3984	0.3742
	80	0.4580	0.4735
	100	0.4277	0.4717
SWF(20 steps)	60	0.5098	0.5221
	80	0.5055	0.5159
	100	0.4024	0.4359

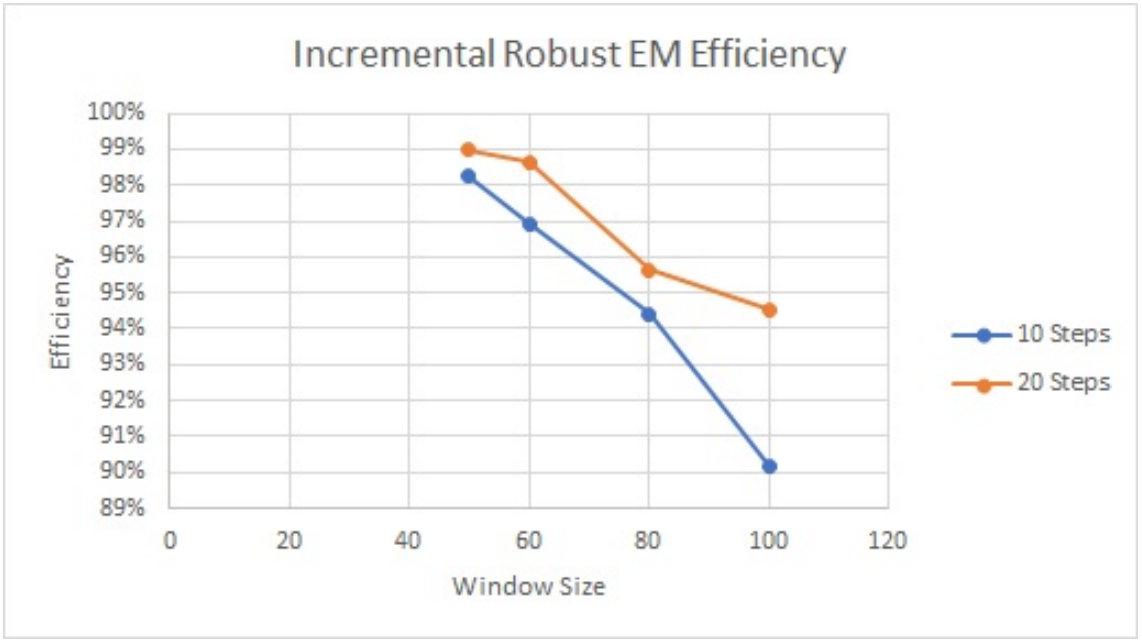

FIGURE 5.3: The computational efficiency factor curve for different window sizes and update periods.

5.6.1 Bias Effect

The RMSE comparison with and without bias inclusion after outlier rejection from simulation is provided in Table 5.2. The results indicate the accuracy has slightly improved with taking into account the bias. On the other hand, an increase in the computational cost was observed as expected. A profile of the accelerometer bias (x-axis) with uncertainty from simulation dataset is shown in Figure 5.4.

TABLE 5.2: The RMSE comparison with and without bias inclusion after outlier rejection (simulation).

Window size	Type	RMSE(m)	RMSE(Deg)
60 (10 steps)	With bias	0.3633	0.3511
	Without bias	0.3677	0.3511
80 (10 steps)	With bias	0.3970	0.4530
	Without bias	0.4780	0.5088

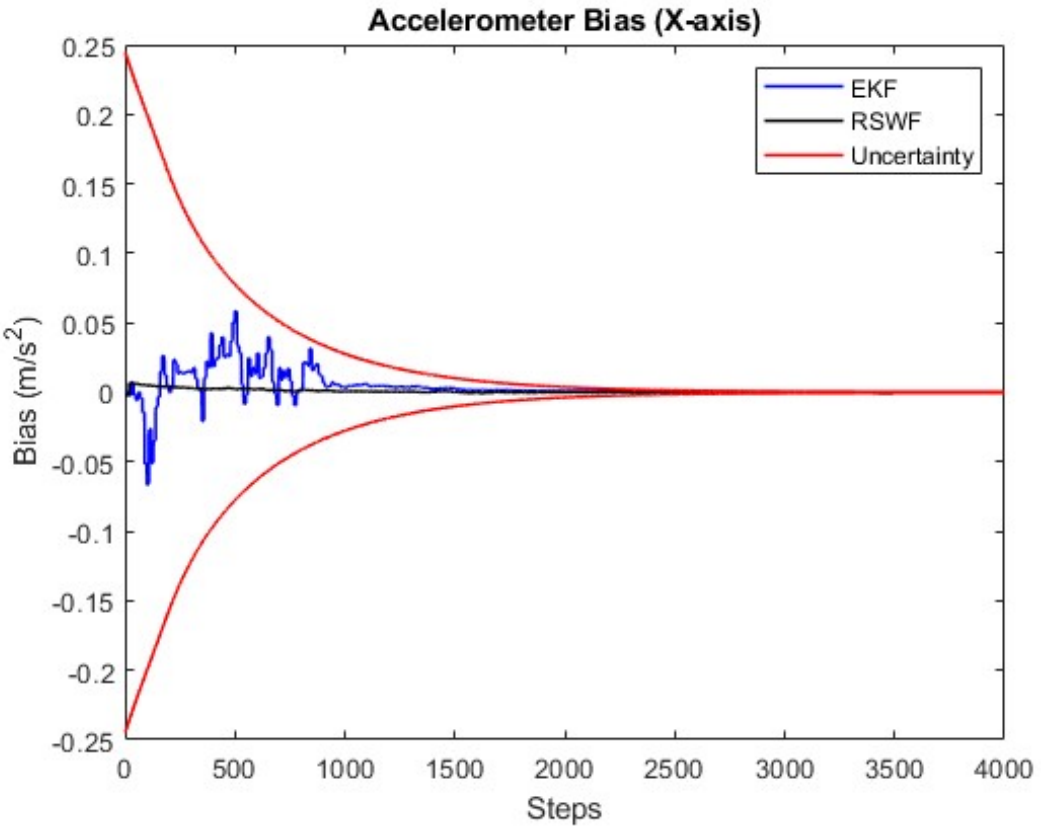

FIGURE 5.4: The Accelerometer bias profile for EKF and RSWF with uncertainty using simulation data.

5.7 Experimental Results

This section presents the RSWF results using the experimental dataset. The position and orientation smoothed trajectories after outlier rejection are manifested in Figure 5.5 where inliers and outliers are shown as green and red dots respectively. The figure shows the RSWF results after two iterations where the outliers or weights of all observations have converged. Unlike, the robust-BMFLS that required six iterations for the same time period, the RSWF has a better outlier detection convergence rate.

Further, the Figures 5.6 and 5.7 provide a comparison between the RSWF results (shown in blue), the EKF output (black) and the robust-BMFLS (orange). We can observe the EKF trajectory is impacted by the outlier presence (e.g $0-6s$) while the RSWF trajectory is running through the captured inlier observations.

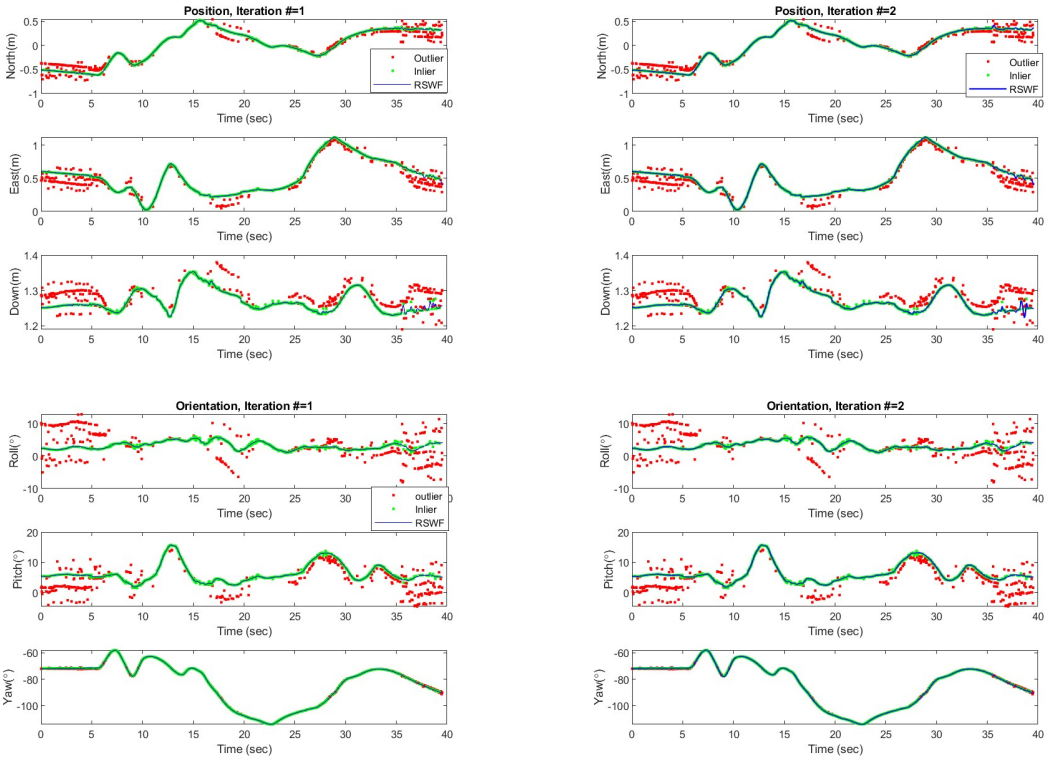

FIGURE 5.5: The RSWF (blue line) results after two iterations where outliers (red) have converged.

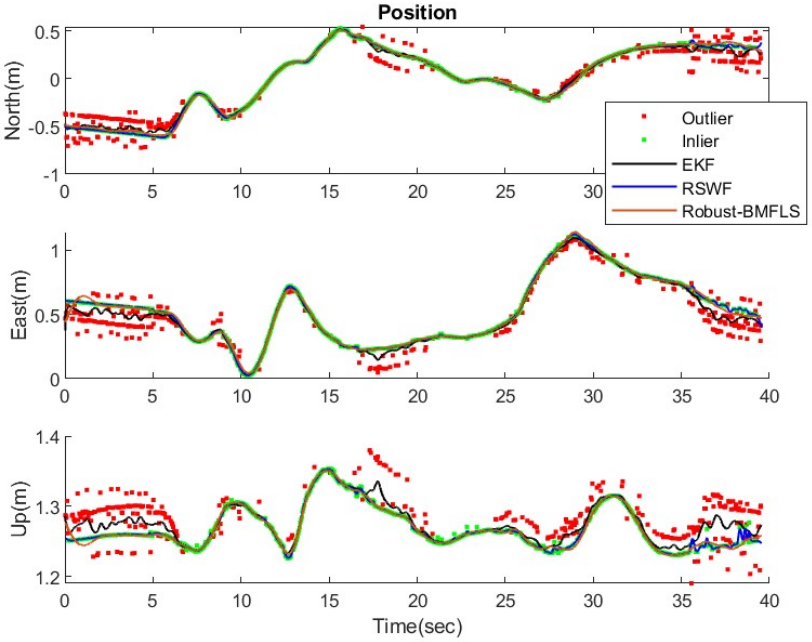

FIGURE 5.6: The RSWF smoothed position trajectory shown in blue after classification of inliers (green) and outliers (red).

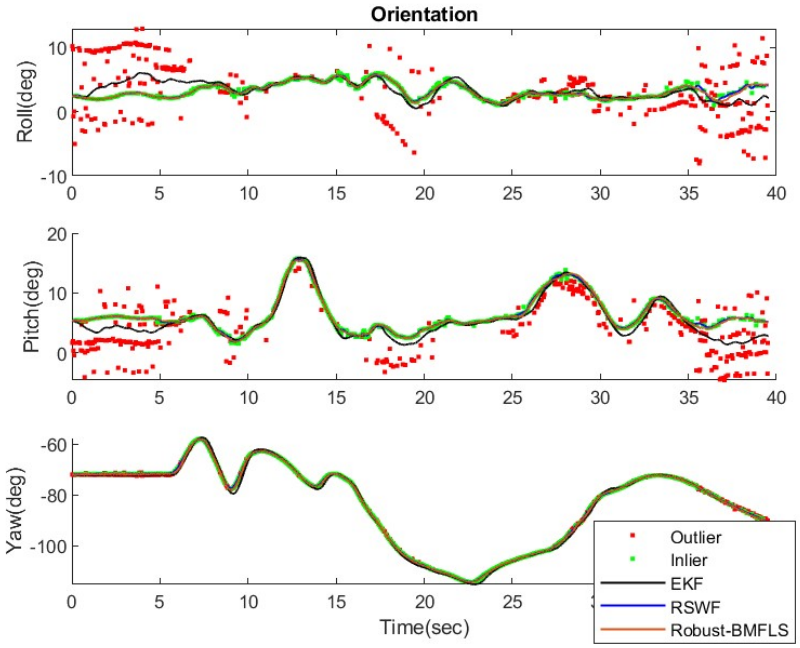

FIGURE 5.7: The RSWF smoothed orientation trajectory shown in blue after classification of inliers (green) and outliers (red).

A better comparison visualisation is shown using the projected 2D trajectory in Figure 5.8 where the EKF estimation (black line) is impaired by erroneous measurements. Instead, the robust EM approach with sliding window (or RSWF) seems to give an improved estimation ability. The 3D raw position vision measurements for 10000 steps is shown in Figure 5.9. Additionally, the 3D RSWF smoothed trajectory after outlier rejection is shown in Figure 5.10. It is evident the impact of outliers on the smoothed trajectory are reduced. The numerical results are analysed in the next few sections.

5.7.1 Window Size and Comparison with Full-Batch

As there is no ground-truth in the experimental dataset and to make a comparison with full-batch estimation, the root mean-square error (RMSE) is calculated using the captured inlier observations as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(z_{k_{inlier}} - \hat{x}_k)^2}. \tag{5.23}$$

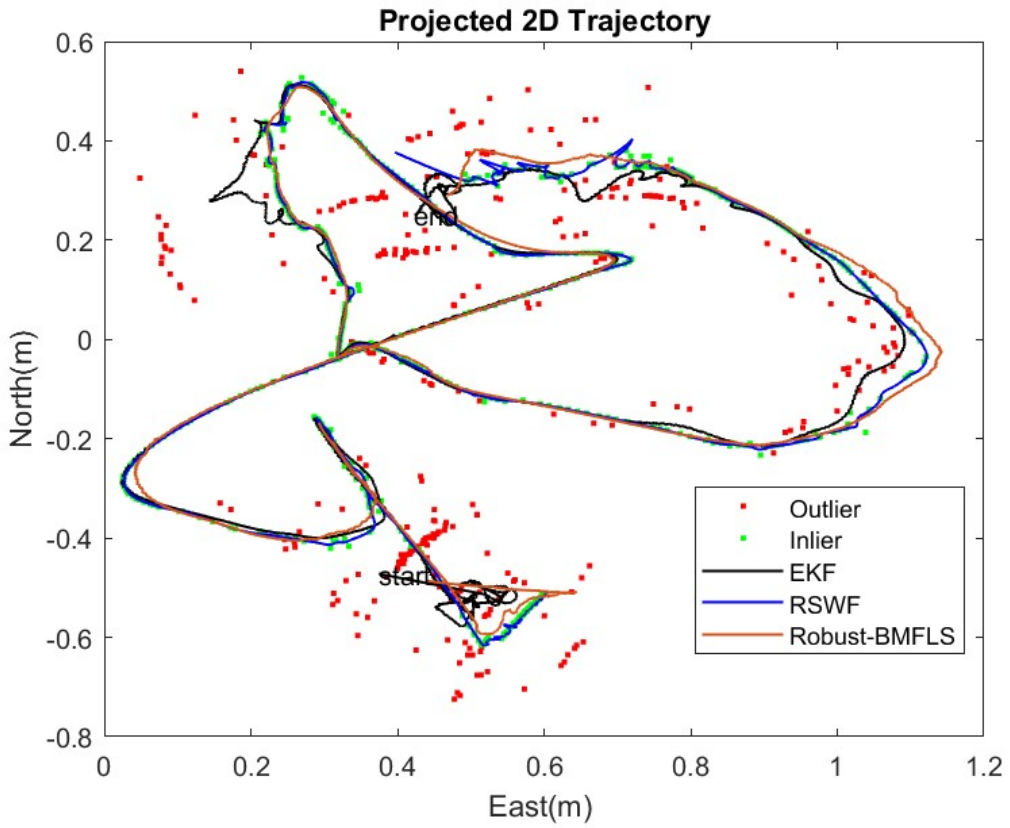

FIGURE 5.8: The projected vehicle 2D trajectory of the RSWF with comparison to the EKF and robust-BMFLS.

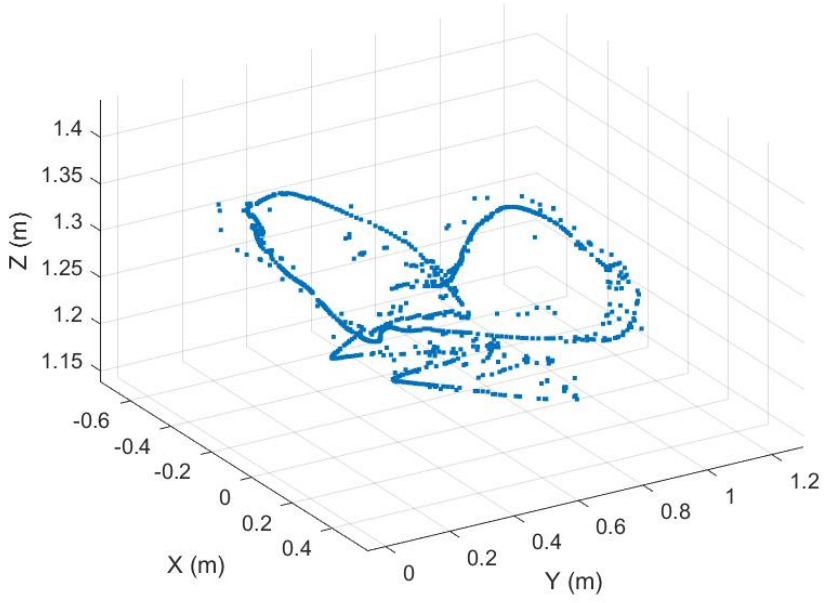

FIGURE 5.9: A proportion of the 3D position raw vision measurements.

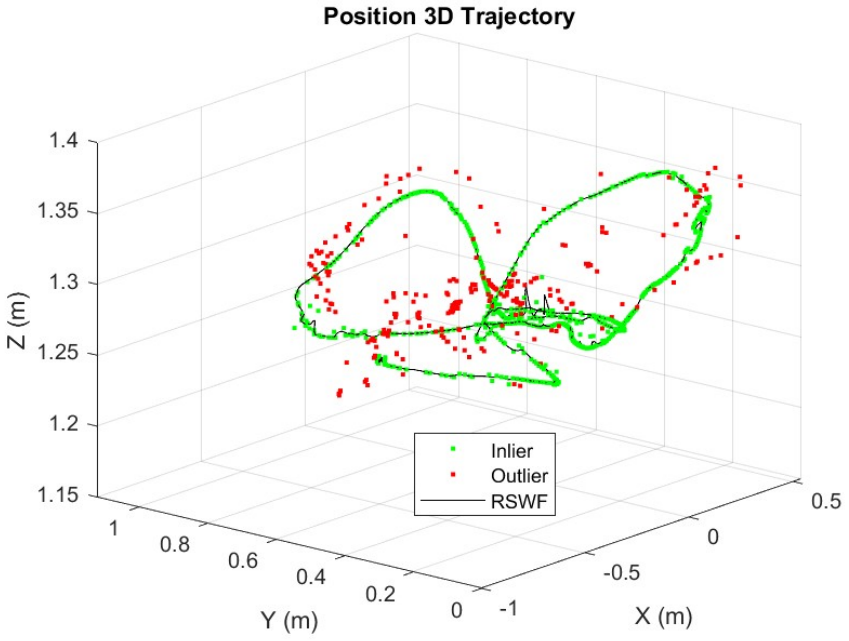

FIGURE 5.10: The 3D smoothed trajectory shown in black line after classification of inliers (green) and outliers (red).

A summary of the different window sizes performance for 1 step marginalisation are shown in Tables 5.3 and 5.4. It can be observed that there is loss in accuracy as the window size is reduced while the full batch estimation yields the lowest RMSE after outlier rejection. On the other hand, a large window size increases the computational cost. The plots of RMSE variation with time steps for translational and rotational motions are in Figure 5.11.

TABLE 5.3: Transational RMSE comparison for different window sizes after outlier rejection (experimental).

Method	Window size	RMSE(m)
Full Batch Est	N/A	3.5108e-04
SWF	60	1.6e-03
	80	1.5e-03
	100	1.3e-03

Further, Figure 5.12 illustrates the RSWF RMSE plots for different window sizes and periodic updates. The largest window size (80) with periodic update of every 10 steps has resulted in lowest RMSE. Conversely, the window size (60) with periodic update of 20 steps has the largest RMSE. The main point here is understanding the accuracy loss for

TABLE 5.4: Rotational RMSE comparison for different window sizes after outlier rejection (experimental).

Method	Window size	RMSE(rad)
Full Batch Est	N/A	5.1916e-04
SWF	60	9.5622e-04
	80	7.0716e-04
	100	7.2035e-04

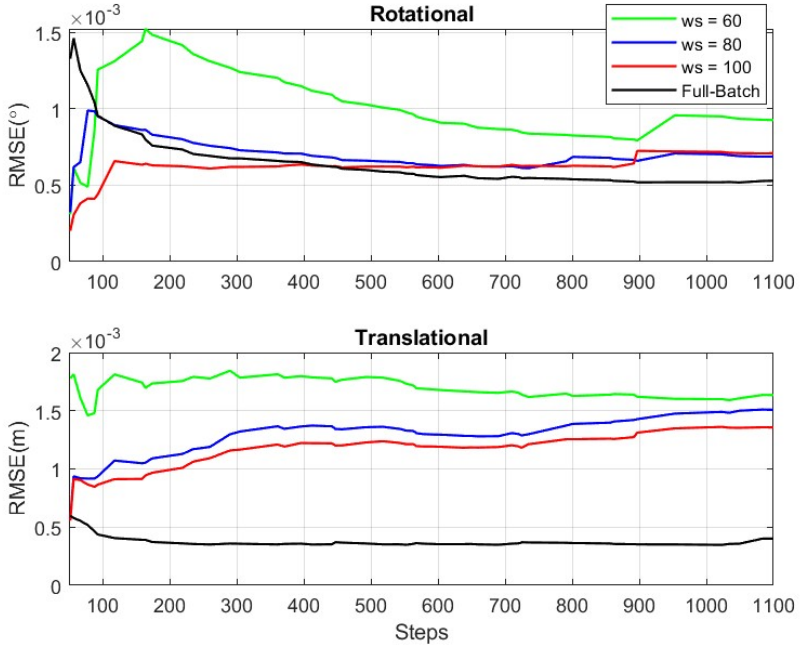

FIGURE 5.11: The RMSE plots with different window sizes (1 step update) using experimental data after outlier rejection for rotational and translational motions.

different window sizes and periodic updates which is further analysed in the next section.

5.7.2 Marginalisation Effect

In addition, the marginalisation size is studied to evaluate the accuracy for different cases. Here, the full-batch estimation is used as the baseline. Thus, the RMSE formulae becomes:

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(\hat{x}_{k_{fullbatch}} - \hat{x}_{k_{SWF}})^2}. \tag{5.24}$$

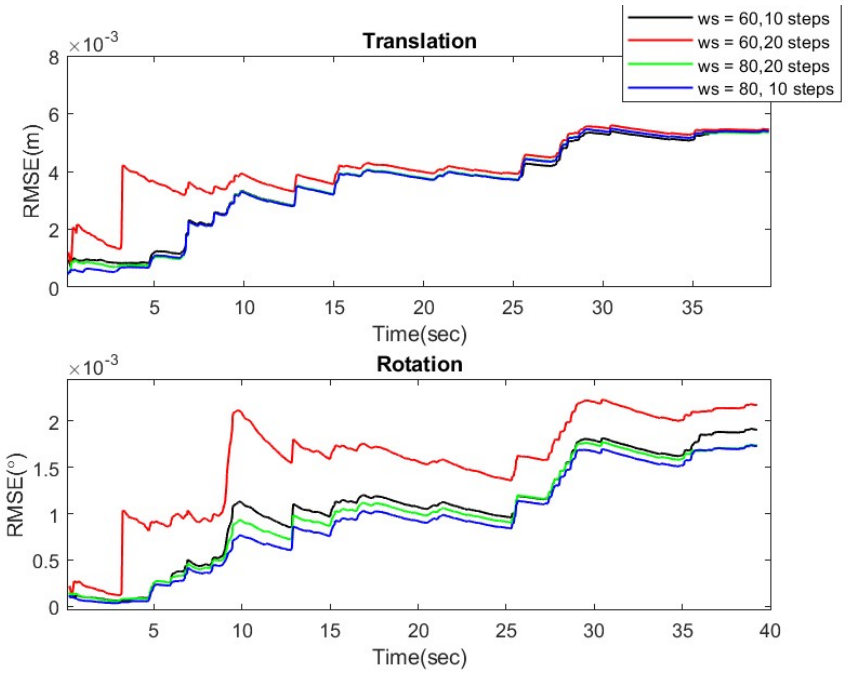

FIGURE 5.12: The RMSE comparison for different window sizes and periodic updates (experimental).

The translational and rotational RMSE is summarised in Table 5.5. The results indicate the notion of periodic updates is useful. For instance, for window size 60, the accuracy lost between updating the states at every 10 and 20 steps is insignificant. Hence, updating at every 20 steps is a wise choice.

TABLE 5.5: The RMSE comparison for different window sizes and update periods after outlier rejection using experimental dataset for 2000 steps.

Method	Window size	RMSE(m)	RMSE(rad)
SWF(10 steps)	60	0.0046	0.0011
	80	8.416e-04	5.485e-04
	100	3.884e-04	2.998e-04
SWF(20 steps)	60	0.0062	0.0015
	80	0.001	8.039e-04
	100	4.699e-04	4.545e-04

5.7.3 Robust-BMFLS and RSWF Comparison

The performance of the robust-BMFLS and RSWF are compared in this section. Table 5.6 depicts the computational run-time ratio of RSWF compared to the robust-BMFLS.

The RSWF run-time is roughly 67% (window size = 60) and 32% (window size = 80) higher than robust-BMFLS for 1 step update. But, the computational burden is overcome by performing periodic updates in RSWF. Here, the relative run-time for periodic updates at every 10 or 20 steps have dropped significantly compared to 1 step update rate.

Moreover, Figure 5.13 is the RMSE curves for the two approaches. It is evident the RSWF has yielded a lower RMSE compared to robust-BMFLS. Thus the RSWF outperforms robust-BMFLS accuracy-wise.

TABLE 5.6: The relative run-time using experimental data for each method for 1 step update.

Window/Lag Size	Update Rate	Robust-BMFLS	RSWF
60	1	1	1.67
80	1	1	1.32
60	10	1	0.651
80	10	1	0.584
60	20	1	0.318
80	20	1	0.290

Duration = 40s, number of steps is 10000.

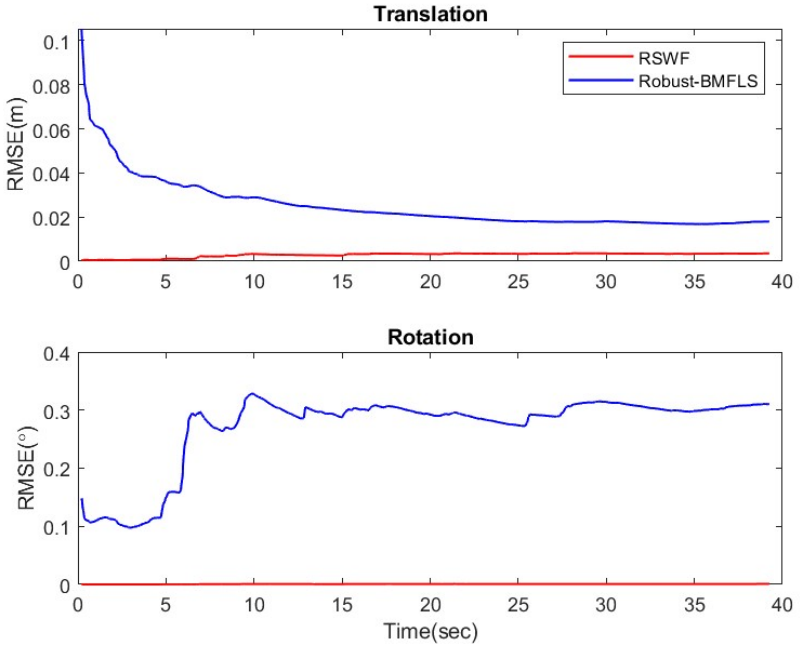

FIGURE 5.13: The RMSE plots for the robust-BMFLS and the RSWF with window/lag size of 80 steps.

5.8 Discussion

The simulation study and experimental results in this chapter have illustrated the RSWF. The experimental results depict the incremental robust EM method is capable of handling a high cluster of outlier presence such as in intervals (0-6) seconds and (35-40) seconds in the experimental dataset (Figures 5.6 and 5.7). It is also shown EKF is vulnerable to wrong measurements.

One attribute in this investigtion which existing work have not been considered is the visual distinction of outlier (red) and inlier (green) measurements as shown in the results in Chapter 4 and 5 which makes clear for an end-user to evaluate and study the work. The effect of optimisation window size and different update periods on the accuracy and performance run-time are important points that are discussed.

From simulation and experiment, it is evident the window size has a direct impact on the navigation performance. Full-batch estimation which iterates over the entire trajectory gives lowest RMSE. While, decreasing the optimisation window size means accuracy loss.

Further, it become evident that full-batch estimation outperforms accuracy-wise, it becomes computationally intractable with time and thus not suitable for close real-time applications. Rather, the incremental robust has eased the computational burden. In particular, the notion of performing period updates at (e.g every 10 or 20 steps) has proved a substantial reduction in the computational cost with negligible loss in accuracy as shown using simulation and experimental data. One suggestion that could further ease the computational burden in SWF is the notion of smart marginalisation mentioned [15] that maybe further explored.

In comparison, the RSWF solution stems a better accuracy than the iterative smoothing and outlier removal presented in Chapter 4. The RSWF has a better outlier detection capability in intervals with intense number of outliers and faster convergence rate. Whereas, the methodology in Chapter 4, requires more iterations in such case. On the other hand, the outlier removal mechanism in Chapter 4 makes it convenient to implement as compared to the solution in this chapter and the literature review. But, by implementing the notion of periodic updates in RSWF has further eased the computational run-time.

Finally, based on the degree of accuracy required and the computing power available, the end user could select the optimum window size and update rate to achieve the desired outcome.

5.9 Summary

This chapter has presented the incremental robust solution with EM (or RSWF). The algorithm was verified using simulation and experimental dataset. Various window sizes with different update periods are studied where accuracy and computational achieved were compared. The RSWF found to be capable of identifying high cluster of outliers with an improved efficiency compared to full-batch estimation. Lastly, updating the states in the incremental solution at lower rate (e.g every 10 or 20 steps) has showed a significant improvement in computational run-time.

Chapter 6

Conclusions and Future Considerations

6.1 Overview

The focus of this thesis was to develop online robust and enhanced solutions for 6-DoF IMU-vision based underwater navigation. This has been illustrated by developing two separate state estimation approaches filtering (EKF) and nonlinear least-squares optimisation (NLS). The outlier detection and removal methods in Chapter 4 and 5 showed a significant enhancement in the navigation trajectory in the experimental dataset.

We saw the proposed smoothing based EKF (robust-BMFLS) approach is easy to implement, but its limitation is it requires more iteration to converge. The problem was then further investigated using optimisation. The RSWF solution found to outperform the robust-BMFLS accuracy-wise. By transforming the robust full-batch estimation to RSWF, an online solution is achieved. However, by introducing the concept of periodic updates in RSWF resulted in substantial reduction in computational requirements (Section5.7.3).

Finally, ROVs operate in a difficult environment and knowing its pose in the global frame is important. Thus, robust navigation algorithms are beneficial in intervention missions for ROVs to complete a task successfully.

6.2 Summary of Contributions

A summary contribution of this thesis include:

- Developing a robust and efficient approach using Chi-square test (Mahalanobis gating test) to mitigate the impact of outliers on navigation accuracy;
- By combining various techniques, an incremental robust navigation solution with EM was found using NLS optimisation;
- Introducing an efficient RSWF solution to further ease the computational cost in optimisation with insignificant accuracy loss.

6.3 Future Considerations

The future investigations that are valuable for enhancing the robustness and efficiency for this work or state estimation techniques are:

- Investigating the problem with particle filter based methods [40],
- Considering an adaptive solution in both solutions presented. For instance, in regions with few or no outliers, a smaller window or lag size is used,
- Developing a compressed-smoothing navigation filter where the bias state could be updated less frequently compared to the essential states (position, velocity and orientation), similar to the notion in the compressed-SLAM [17].

Bibliography

[1] Theophile Cantelobre, Clement Chahbazian, Arnaud Croux, and Silvere Bonnabel. A real-time unscented kalman filter on manifolds for challenging auv navigation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2309–2316. IEEE, 2020. ISBN 9781728162126.

[2] Francisco Bonin-Font, Gabriel Oliver, Stephan Wirth, Miquel Massot, Pep Lluis Negre, and Joan-Pau Beltran. Visual sensing for autonomous underwater exploration and intervention tasks. *Ocean engineering*, 93:25–44, 2015.

[3] Arturo Gomez Chavez, Christian A Mueller, Tobias Doernbach, and Andreas Birk. Underwater navigation using visual markers in the context of intervention missions. *International journal of advanced robotic systems*, 16(2), 2019.

[4] Dinh Van Nam and Kim Gon-Woo. Robust stereo visual inertial navigation system based on multi-stage outlier removal in dynamic environments. *Sensors (Basel, Switzerland)*, 20(10):2922–, 2020.

[5] Stefan Bernard Williams. *Efficient Solutions to Autonomous Mapping and Navigation Problems*. PhD thesis, 2001-01-01. URL `http://hdl.handle.net/2123/809`.

[6] Mohinder S. Grewal. *Kalman filtering : theory and practice*. Prentice-Hall information and system sciences series. Prentice-Hall, Englewood Cliffs, N.J, 1993.

[7] Zoran Sjanic, Martin A. Skoglund, Thomas B. Schön, and Fredrik Gustafsson. A nonlinear least-squares approach to the slam problem*. *IFAC Proceedings Volumes*, 44(1):4759–4764, 2011. ISSN 1474-6670. 18th IFAC World Congress.

[8] Simo Särkkä. *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2013. doi: 10.1017/CBO9781139344203.

[9] Xiang Gao and Tao Zhang. *Introduction to Visual SLAM: From Theory to Practice*. Springer Singapore Pte. Limited, Singapore, 2021. ISBN 9789811649387.

[10] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, 2004.

[11] Timothy D. Barfoot. *Nonlinear Non-Gaussian Estimation*, page 88–144. Cambridge University Press, 2017. doi: 10.1017/9781316671528.005.

[12] Hong Xu, Keqing Duan, Huadong Yuan, Wenchong Xie, and Yongliang Wang. Adaptive fixed-lag smoothing algorithms based on the variational bayesian method. *IEEE transactions on automatic control*, 66(10):4881–4887, 2021. ISSN 0018-9286.

[13] Mohinder S Grewal and Angus P Andrews. Optimal smoothers. In *Kalman Filtering*, Wiley - IEEE, pages 239–279. Wiley, Hoboken, NJ, USA, 4 edition, 2014. ISBN 9781118851210.

[14] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. *A Sliding Window Filter for Incremental SLAM*, pages 103–112. Springer US, 2008. ISBN 978-0-387-75523-6.

[15] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *Journal of field robotics*, 27(5):587–608, 2010. ISSN 1556-4959.

[16] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE transactions on robotics*, 24(6):1365–1378, 2008.

[17] Jonghyuk Kim, Jose Guivant, Martin L. Sollie, Torleiv H. Bryne, and Tor Arne Johansen. Compressed pseudo-slam: pseudorange-integrated compressed simultaneous localisation and mapping for unmanned aerial vehicle navigation. *Journal of navigation*, 74(5):1091–1103, 2021. ISSN 0373-4633.

[18] Charu C. Aggarwal. *Outlier Analysis*. Springer International Publishing, 2nd ed. 2017. edition, 2017.

[19] Gabriel Agamennoni, Juan I Nieto, and Eduardo M Nebot. An outlier-robust kalman filter. In *2011 IEEE International Conference on Robotics and Automation*, pages 1551–1558. IEEE, 2011.

[20] Guobin Chang. Robust kalman filtering based on mahalanobis distance as outlier judging criterion. *Journal of geodesy*, 88(4):391–401, 2014.

[21] Thomas Kautz and Bjoern M Eskofier. A robust kalman framework with resampling and optimal smoothing. *Sensors (Basel, Switzerland)*, 15:4975–4995, 2015. ISSN 1424-8220.

[22] Ali S Hadi. Identifying multiple outliers in multivariate data. *Journal of the Royal Statistical Society. Series B, Methodological*, 54(3):761–771, 1992.

[23] Kyuman Lee and Eric N Johnson. Robust outlier-adaptive filtering for vision-aided inertial navigation. *Sensors (Basel, Switzerland)*, 20(7):2036–, 2020. ISSN 1424-8220.

[24] Larry J. Stephens. *Schaum's Outline of Theory and Problems of Beginning Statistics*. Schaum's Outline Series McGRAW-HILL. McGraw-Hill, Englewood Cliffs, N.J, 1998.

[25] Vinay A. Bavdekar, Anjali P. Deshpande, and Sachin C. Patwardhan. Identification of process and measurement noise covariance for state and parameter estimation using extended kalman filter. *Journal of process control*, 21(4):585–601, 2011. ISSN 0959-1524.

[26] Zoran Sjanic, Martin A. Skoglund, and Fredrik Gustafsson. Em-slam with inertial/visual applications. *IEEE transactions on aerospace and electronic systems*, 53(1):273–285, 2017. ISSN 0018-9251.

[27] Michael George and Salah Sukkarieh. Tightly coupled ins/gps with bias estimation for uav applications. *Proceedings of the 2005 Australasian Conference on Robotics and Automation, ACRA 2005*, 01 2005.

[28] J Russell Carpenter and Christopher N D'Souza. *Navigation Filter Best Practices*. NASA/Langley Research Center, Hampton, 2018.

[29] Jo-Anne Ting, Evangelos Theodorou, and Stefan Schaal. Learning an outlier-robust kalman filter. In *Machine Learning: ECML 2007*, Lecture Notes in Computer Science, pages 748–756, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 9783540749578.

[30] Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Robust pose-graph loop-closures with expectation-maximization. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 556–563, 2013. doi: 10.1109/IROS.2013.6696406.

[31] Jiantong Cheng, Jonghyuk Kim, Jinliang Shao, and Weihua Zhang. Robust linear pose graph-based slam. *Robotics and autonomous systems*, 72:71–82, 2015. ISSN 0921-8890.

[32] Dongdong Peng, Tian Zhou, John Folkesson, and Chao Xu. Robust particle filter based on huber function for underwater terrain-aided navigation. *IET radar, sonar navigation*, pages 1867–1875, 2019. ISSN 1751-8784.

[33] Hyon Lim and Young Sam Lee. Real-time single camera slam using fiducial markers. In *2009 ICCAS-SICE*, pages 177–182. IEEE, 2009. ISBN 9784907764340.

[34] Michail Kalaitzakis, Brennan Cain, Sabrina Carroll, Anand Ambrosi, Camden Whitehead, and Nikolaos Vitzilaios. Fiducial markers for pose estimation: Overview, applications and experimental comparison of the artag, apriltag, aruco and stag markers. *Journal of intelligent robotic systems*, 101, 2021. ISSN 0921-0296.

[35] Oliver J.Woodman. An introduction to inertial navigation. Technical report, University of Cambridge, 2007.

[36] Brenton Lee Leighton. Accurate 3d reconstruction of underwater infrastructure using stereo vision. Master's thesis, 2019.

[37] Fredrik Gustafsson. Particle filter theory and practice with positioning applications, 2010. ISSN 0885-8985.

[38] Dingjie Wang, Yi Dong, Qingsong Li, Zhaoyang Li, and Jie Wu. Using allan variance to improve stochastic modeling for accurate gnss/ins integrated navigation. *GPS solutions*, 22:1–14, 2018.

[39] Robert Grover Brown. *Introduction to random signals and applied Kalman filtering : with MATLAB exercises*. John Wiley, 4th ed. edition, 2012.

[40] Branko. Ristic, Sanjeev. Arulampalm, and Neil. Gordon. *Beyond the Kalman filter particle filters for tracking applications*. Artech House Boston.London, 2004. ISBN 1-58053-631-X.

[41] Chong-hyok Kim. *Autonomous navigation for airborne applications*. PhD thesis, 2004.

[42] Liam Paull, Sajad Saeedi, Mae Seto, and Howard Li. Auv navigation and localization: A review. *IEEE journal of oceanic engineering*, 39:131–149, 2014. ISSN 0364-9059.

[43] Murray R Spiegel, John J Schiller, and R. Alu Srinivasan. *Schaum's outline of probability and statistics*. McGraw-Hill's AccessEngineering. McGraw-Hill, 4th ed. edition, 2012. ISBN 9780071795579.

[44] Hamid Ghorbani. Mahalanobis distance and its application for detecting multivariate outliers. *Facta universitatis. Series, mathematics and informatics*, pages 583–, 2019. ISSN 0352-9665.

[45] Shoudong Huang, Yingwu Lai, U Frese, and G Dissanayake. How far is slam from a linear least squares problem? In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3011–3016. IEEE, 2010. ISBN 9781424466740.

[46] Chong-hyok Kim. *Lecture notes in Robotics, Kalman Filter*. University of Technology Sydney, Spring, 2021.

[47] Chong-hyok Kim. *Lecture notes in Robotics, Extended Kalman Filter and SLAM*. University of Technology Sydney, Spring, 2021.