Interactive Multiple Model Filtering for Robotic Navigation and Tracking Applications

by

©Patrick Joseph Glavine

A Thesis submitted to the School of Graduate Studies in partial fulfilment of the requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

October 2019

St. John's Newfoundland

Abstract

The work contained in this thesis focuses on two main objectives. The first objective is to evaluate the Interactive Multiple Model (IMM) filter method for robotic applications including inertial navigation systems (INS) and computer vision tracking. The second objective is to design an experimental testbed for multi-model mobile robot state estimation research in the Intelligent Systems Laboratory (ISLAB) at Memorial University.

An IMM estimator uses multiple filters that run simultaneously to produce a combined weighted estimation of an observed system's states. The weights are functions of the likelihood of how well each individual filter matches the current behaviour exhibited by the system. The performance of IMM filtering is evaluated using two different strategies for augmenting the system's filter banks. The first method uses multiple kinematic models (constant velocity and constant acceleration models) in a mean-shift-based computer vision tracking application. The results of this experiment indicate that the IMM improves tracking performance due to its ability to adapt to the continuously changing motion characteristics of 2D blobs in videos. The second approach uses the same kinematics for each filter; however, the process and sensor noise parameters are tuned differently for each model. This method is tested in INS applications for both an automobile and a skid-steer mobile robot (Seekur Jr). Results show that the method improves INS tracking over single model Extended Kalman Filter (EKF) designs. Furthermore, an augmented state-space model containing skid-steer instantaneous center of rotation (ICR) kinematics is presented for future testing on the Seekur Jr INS.

The experimental testbed designed in this thesis work is an operational data acquisition system developed for use with the Seekur Jr robot. The Seekur Jr platform has been Robot Operating System (ROS) enabled with access to data streams from 2D Lidar, 3D nodding Lidar, inertial measurement unit, digital compass, wheel encoder, onboard Global Positioning System (GPS), real-time kinematic (RTK) differential global positioning system (DGPS) ground truth, and vision sensors. The physical setup and data networking aspects of the testbed have been used for validation of an IMM filter presented in this thesis and is fully configured for future multi-model localization experiments of the ISLAB.

Acknowledgements

Without the help of the following individuals, the completion of this Master of Engineering degree would not have been possible. My supervisors, Dr. Oscar De Silva, Dr. George Mann and Dr. Raymond Gosine, thank you for the unwavering support throughout the entirety of my master's program. The knowledge and expertise that you provided were invaluable assets every step of the way. Your patience, dedication and enthusiasm towards my learning and development as a researcher were essential for my successes. It was an honour and a privilege to work with you at the Intelligent Systems Laboratory at Memorial University.

My labmates Mr. Ravindu Thalagala, Mr. Eranga Fernando, Dr. Trung Nguyen and Dr. Thumeera Wanasinghe, thank you all for your constant support and assistance throughout this program. You were all a pleasure to work with.

Our engineering co-op work term student Mr. Matthew King, thank you for your exceptional help with configuring the communication and control network for the Seekur Jr experimental setup. Your work facilitated the entire testbed design process.

To my supportive family, thank you for always believing me and encouraging me to keep pursuing my interests. Without you, this accomplishment would never have been possible.

This research was supported by funding from the National Sciences and Engineering Research Council of Canada (NSERC) and Memorial University of Newfoundland, Canada.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ARIA	Advanced Robot Interface for Applications
CA	Constant Acceleration (Model)
CPU	Central Processing Unit
CV	Constant Velocity (Model)
DGPS	Differential Global Positioning System
EKF	Extended Kalman Filter
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
ICR	Instantaneous Center of Rotation
IMM	Interactive Multiple Model
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IP	Internet Protocol
ISLAB	Intelligent Systems Laboratory
KLT	Kanade-Lucas-Tomasi
MCL	Monte Carlo Localization
MHE	Moving-Horizon Estimation
NED	North-East-Down
PDF	Probability Density Function

RMS	Root Mean Square
ROS	Robot Operating System
RTK	Real-Time Kinematic
SBAS	Satellite-Based Augmentation System
SLAM	Simultaneous Localization and Mapping
SPS	Standard Positioning Service
SSH	Secure Shell
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver-Transmitter
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
WGS84	World Geodetic System 84

Chapter 1

Introduction

About this chapter: This chapter discusses autonomous system technologies and introduces typical methods used for state estimation. The Kalman filter and nonlinear variations are introduced, as well as the interactive multiple model filter, which is the main filtering technique investigated in this thesis. The chapter introduces the thesis problem statement and the main objectives of this Master of Engineering research project.

1.1. Introduction

Improvements in computing technology and the rapid development of intelligent control systems has led to the integration of autonomous technologies in most industries worldwide. Examples of these technologies include robotic manufacturing equipment [1], self-piloting unmanned-aerial vehicles (UAVs) [2] and planetary rovers for space exploration [3]. Incorporating autonomous systems in engineering or commercial settings can improve task efficiency, ensure repeatable work precision and eliminate the risk of human injury in dangerous environments.

The growth of autonomous technologies has been facilitated by advancements in integrated circuit design for sensors, computing devices, and intelligent control algorithms [4]. Improved central-processing unit (CPU) architectures provides the required power to implement sophisticated software packages for autonomous systems. The increased CPU

computing speed allows large datasets to be quickly processed for real-time use in software applications. This has been a driving force for the development of vision-based systems that rely on high-resolution cameras for control-loop feedback [5]. The development of advanced sensing devices has ensured that autonomous systems are provided with consistently precise measurement information and minimizes the influence of noise corruption on sensor signals [4].

Most autonomous systems have two main processes that must be completed simultaneously during typical operations. The first process is state estimation; the system must use the available sensor information and control system inputs to determine the current state of the system (i.e.: robotic end-effector position, velocity and orientation) [6]. The second process is control; given the current state of the system, the next control inputs required to reach the goal state (i.e.: move robotic end-effector from the current position to the workpiece) must be determined [7].

The research work contained in this thesis studies the state estimation problem for autonomous systems. One of the most globally popular state estimation techniques used in many engineering applications is known as the Kalman filter [6]. This algorithm is best suited for linear time-invariant systems [8]. The algorithm assumes a stochastic system model with noise-corrupted measurements [6]. It optimizes estimation performance by adaptively adjusting the estimator gain in response to the changing mean-squared error of the state covariance [6]. The Kalman filter is an effective estimation method when the system is linear, and the dynamic model is well-defined [7].

The Extended Kalman Filter (EKF) addresses the problem of nonlinearity in the system model [7]. The algorithm linearizes the system dynamic equations about an operating point that is defined by the most recent state estimate [6]. The EKF is then implemented in the same way as the Kalman filter using the nonlinear system equations to predict the next set of system states [7]. For highly nonlinear systems, the EKF can show unstable behaviour and produce high errors due to abrupt changes in system states near the linearized operating point [9]. When this occurs, the estimator can quickly diverge from the true state of the system. This issue is handled by another version of the Kalman filter known as the unscented Kalman filter (UKF). The UKF method uses the unscented transform to propagate sample points through the nonlinear function to produce a Gaussian approximation of the function [10]. This method, however, can be slower than the EKF in practical applications [11].

One of the common disadvantages of the Kalman filter, EKF and UKF is their limitation of having a single dynamic model for state estimate propagation. In many cases, the behaviour of a system varies depending on several possible factors including abrupt changes in the control inputs or arbitrary system interactions with external surroundings. Examples of these include aggressive turning manoeuvres made by an aircraft [12] or an autonomous ground vehicle skidding/slipping laterally across a surface [13]. Using a single system model may not account for these dynamic changes, therefore, including multiple models may improve an estimator's ability to maintain accurate state tracking [14].

There are several different strategies for incorporating multiple system models in an estimator. Among which, the interactive multiple model (IMM) algorithm is a preferred

and established method used in many aircraft target tracking applications [15] [16] [17] [18]. Recent work in [19] [20] [21], investigates its application for vehicle navigation systems. The IMM filter addresses the multiple model estimation problem by running a bank of filters simultaneously in parallel and combining the estimates of each filter using weighted probabilities [14]. The probabilities are recursively calculated by the filter and represent the likelihood of how well the models each capture the current dynamic behaviour that the system is exhibiting [19]. The main disadvantage of this filtering technique is its suboptimality due to the estimates being a mixed result from multiple possible models. However, if the models included in the IMM design are limited to realistic candidates that capture the expected system dynamics and their uncertainties, then these errors can be minimized and the benefits of using multiple models can improve tracking results [14]. The Intelligent Systems Laboratory (ISLAB) of Memorial University of Newfoundland is currently developing multiple model navigation techniques for its robotic fleet comprised of a Seekur Jr, Pioneer robots and micro aerial platforms. The long-term objective of this research group is developing reliable fleet operations for missions that have changing operating conditions. This thesis evaluates the IMM filter for this purpose by designing filtering banks to effectively capture the operating modes and uncertainties of robotic tracking and localization applications. The thesis first evaluates the IMM strategy for a computer vision tracking problem to validate algorithm performance. The IMM method is then implemented and validated for vehicle localization using the KITTI Vision Benchmark dataset [22]. The IMM filter design is then modified for localization of the Seekur Jr mobile robot [23].

1.2. **Problem Statement**

The objective of this research work is to evaluate the effectiveness of IMM filter integration in robotic navigation and tracking applications. The main experiments contained herein are:

- Computer Vision Tracking – Tracking the motion of arbitrary blob targets in video sequences using a colour-based mean shift tracker paired with an IMM filter to improve accuracy. Tests include tracking a constant velocity target, a constant acceleration target, circular motion tracking and general object tracking.
- Automobile Inertial Navigation System (INS) – Designing an INS with an IMM framework using the KITTI Vision Benchmark dataset. The IMM uses differently tuned sets of sensor noise parameters to shift the filter's reliance on each sensor for different driving scenarios (i.e.: driving in a straight line or performing an abrupt turn). The dataset provides inertial measurement unit (IMU), differential global positioning system (DGPS) and orientation measurements. The INS estimates the physical states of the vehicle system such as position, velocity, orientation and sensor biases.
- Skid-Steer Robot INS Design – Designing an INS for a skid-steer mobile robot (Seekur Jr Robot) using an IMM filter. The INS used in the automobile experiment has been redesigned for the Seekur Jr. The varied noise parameter approach is tested experimentally with the robot. The framework for incorporating a skid-steer instantaneous center of rotation (ICR) tracking model in the IMM framework has

been presented and is discussed for future work. The Seekur Jr is equipped with an Emlid Reach module [24] to provide IMU and DGPS measurements for IMM experiments.

1.3. Objectives and Expected Contributions

The focus of this thesis is to evaluate the effectiveness of IMM filtering for robotic navigation and tracking applications. This is achieved by designing and evaluating IMM filters for robotic tracking and navigation problems and developing an experimental testbed for multi-model estimator performance evaluation. The objectives of the thesis are as follows:

Objective 1 – Design an effective computer vision tracking system that implements mean shift and IMM filtering techniques.

Objective 2 – Demonstrate the effectiveness of IMM filtering for automobile INS applications.

Objective 3 – Design an IMM-INS for skid-steer mobile robots using ICR tracking for outdoor navigation applications.

Objective 4 – Develop an experimental testbed for the Seekur Jr robot for multi-model localization research work.

The contributions of this thesis are as follows:

Contribution 1 – IMM design and validation for computer vision target tracking and robotic inertial navigation applications. This evaluates two different strategies for augmenting the model bank of IMM filters (i.e.: models with different process and sensor noise characteristics and models with different system dynamics).

Contribution 2 – Development of an experimental testbed for multiple model estimation based on the Seekur Jr platform. As part of the thesis work the Seekur Jr platform is Robot Operating System (ROS) enabled with access to data streams from 2D Lidar, 3D nodding Lidar, IMU, digital compass, wheel encoder, onboard Global Positioning System (GPS), real-time kinematic (RTK) DGPS ground truth, and vision sensors.

Contribution 3 – Design and experimental validation of an IMM filter for the Seekur Jr mobile robot. The experimental testbed developed in this thesis is used for this purpose.

1.4. Organization of Thesis

The following briefly discusses the contents found in each chapter of this thesis:

Chapter 1 – This chapter introduces the research topics and outlines the objectives of the research work.

Chapter 2 – This chapter discusses related works to this research and provides the necessary theoretical background information regarding existing Kalman filter state estimation techniques including the linear Kalman filter, EKF and IMM. The advantages and disadvantages of these techniques are also discussed.

Chapter 3 – This chapter introduces mean shift theory and its usage in computer vision tracker design. The design process for a two model IMM filter is presented. The vision system is tested on several target tracking scenarios with quantitative analysis and comparisons.

Chapter 4 – This chapter presents the vehicle state space model and measurement model used to design the INS for an automobile. Nonlinear observability analysis for the system is included. The experimental validation of the INS using a two-mode IMM filter is discussed.

Chapter 5 – This chapter introduces skid-steer kinematic models for tracking mobile robot ICRs during operations. The developed experimental platform using the Seekur Jr robot is discussed in detail, including platform design,

sensors used and data processing techniques. The INS is tested using a dataset collected by the Seekur Jr and the results of this experiment are discussed.

Chapter 6 – This chapter presents the conclusions that were drawn from the experiments conducted during this research project. The overall advantages and disadvantages of IMM filtering are discussed with regards to the applications that have been presented. Additional research topics and required work to advance this project further are discussed.

Chapter 2

Background

About this chapter: This chapter reviews existing state estimation techniques that are commonly used in robotic tracking, navigation, and control applications. Kalman filter, EKF and IMM filter theory is discussed in detail to provide the necessary background for understanding the estimators designed in the experiments of chapters 3-5.

2.1. Related Works

2.1.1. State Estimation Techniques and IMM Applications

The area of state estimation for autonomous systems is a rapidly advancing field driven by the work of researchers worldwide. Many methods have been developed over the years for addressing the state estimation problem for various systems. Some standard methods typically employed for tracking and localization tasks include the Kalman filter [8], EKF [6], UKF [9], Monte Carlo localization (MCL) [25], grid-based localization [26], and simultaneous localization and mapping (SLAM) [27].

The Kalman filter provides the optimal solution for stochastic linear time-invariant systems [8]. The algorithm has been modified over the years to solve numerous problems, including systems governed by nonlinear functions. The EKF method uses a first order Taylor series expansion to linearize the nonlinear system equations [6]. The UKF employs

the unscented transform to propagate sample points through the nonlinear function to estimate the mean and covariance of the system states [9]. Although the UKF can perform better than the EKF for highly nonlinear systems, the EKF still remains one of the most widely used Kalman filter formulations for nonlinear state estimation [9].

MCL is a non-parametric localization approach that uses a distribution of weighted samples (particles) to estimate the current and future states of the tracked system given the system inputs and current sensor observations [25]. The samples are recursively propagated forward using the system process model and the sensor information provides corrections to these sample estimates [25]. Successful convergence of this filter occurs when the mean of the particle distribution approaches the true state of the system [7]. MCL can maintain multiple hypotheses for the states of the system and is effective for nonlinear applications [7]. However, if too many samples are used, the algorithm can become computationally expensive, and if too few samples are used, particle deprivation can occur and the filter may not find the solution [7]. Grid-based localization is another effective tracking technique, especially for indoor, structured environment applications. For mobile robot localization, grid-based methods typically require a map that is subdivided into discrete points (grains) [26]. The grains can be assigned an occupancy status to indicate obstacles in the environment [26]. Localization is performed by first propagating the robot states forward using the system motion model [7]. Next, sensor data (i.e.: laser scans) are observed and the algorithm updates its belief states for the robot pose [26]. Like MCL, this algorithm can also maintain multiple hypotheses for the robot pose. Grain coarseness can dictate the effectiveness of this localization method [7]. Fine grains produce accurate

tracking results but cause the process to become computationally expensive while coarse grains improve computation time but reduce accuracy [26].

SLAM is a highly explored area of state estimation due to its applications for robot operations in unstructured environments. The process involves continuously generating and updating a navigation map using landmarks and features detected by onboard sensors while simultaneously using the map to perform localization [27]. Many types of sensors can be used for the SLAM mapping process including cameras [28], radar [29], sonar [30] and laser [31]. One of the main issues with SLAM is the computational cost of processing the large amounts of sensor data [27]. Fortunately, the improvements to computing technologies and to SLAM algorithms in recent years have made implementing these systems progressively more feasible for real-time applications [27].

The IMM filtering method for tracking and localization can be implemented in combination with many of the previously discussed estimation techniques using its model probability mixing framework. For example, the work in [32] implements a three-mode IMM paired with particle filtering for manoeuvring target tracking using only bearing measurements. The modes of the filter include different possible kinematic models that reflect the expected target behaviour [32]. The results of this work demonstrate the high accuracy tracking potential of the IMM approach, however, the high computational load of the particle filter in this experiment was an issue [32]. Another example is the work in [33] which implements an IMM using UKFs with different kinematic models in the filter bank. The results of this implementation show reasonable tracking results with improvements over the single Kalman filter that was compared [33].

The adaptability of the IMM filter makes it a popular choice for manoeuvring target tracking since their motion is generally unpredictable [14]. For this reason, it is a favoured option for tracking and state estimation in the aerospace industry as shown in [15] [16] [17] [18]. Generally, ground vehicles like automobiles or mobile robots have predictable motion trajectories while operating in controlled environments. However, changes in operation terrain or weather conditions can cause unpredictable vehicle movement to occur. Furthermore, aggressive turning manoeuvres made by these systems, especially at high speeds, can lead to sliding/slipping. For these reasons, the IMM method can be a beneficial algorithm to incorporate in typical ground vehicle and mobile robot INS applications. In both [19] and [20], IMMs are designed to address these issues in road vehicle localization. Both papers implement two mode IMMs using EKFs for varying driving conditions. In [19], the first mode models the vehicle kinematic states with no-slip assumptions, while the second mode considers the vehicle dynamics such as lateral forces. The results of this work show that estimates of the kinematic model are more accurate for low-speed operations with low tire slippage, while the dynamic model is more accurate when large tire slippage occurred. When both models are included in the IMM estimator, the vehicle localization becomes more robust and adaptable for the driving conditions [19]. Similarly, in [20] the two modes of the IMM consider different kinematic behaviours of the vehicle. One model is a first-order function for straight driving motion while the other is a second-order equation designed for turning manoeuvres [20]. The findings in [20] reported similar results to [19] indicating that the IMM algorithm is indeed a good candidate for manoeuvring ground vehicle and mobile robot localization tasks. The robotic fleet of the

ISLAB is being developed for missions involving changing operating conditions including indoor-outdoor transitions, kinodynamic model changes of the robots, environmental disturbance level changes, etc. Therefore, the IMM algorithm is deemed a leading candidate to address this long-term objective.

2.1.2. **Localization Experimental Testbeds**

Many research groups have developed experimental testbeds for autonomous system algorithm development. Several of these testbeds have produced datasets that are available online including [34] [22] and [35]. Each of the available datasets contains various combinations of sensing devices for different applications. In many cases, it is difficult to obtain an online dataset that contains all the specific sensor data required for a given localization filter application. This can limit the choices of potential models that can be incorporated into an IMM filter design. Furthermore, the online datasets are for filter design purposes only; control algorithms cannot be evaluated using the available data that these testbeds provide.

The ISLAB at Memorial University has developed several experimental testbed setups for robotic localization and control algorithm testing. Two examples of these testbeds are illustrated in [36] and [37]. In [36] a 3D sensor node for multi-robot localization was designed using an ultrasonic-based range measurement apparatus and infrared camera. The sensor apparatus was evaluated using two Pioneer robots and an aerial robot in the ISLAB. In [37], a multi-robot cooperative localization strategy was tested using

the same two Pioneer robots and the Seekur Jr in the ISLAB. For this experiment, the robots relied on odometry and laser scan measurements for localization.

The experimental testbed designed for this thesis has been developed with the intention of providing a robust platform for future multi-model localization research at the ISLAB. The testbed builds upon the work in [36] and [37] by integrating ROS with the Seekur Jr onboard computer to facilitate sensor configuration and control implementation. The system has been updated to enable outdoor experiments using DGPS and has access to additional sensors including magnetometer, wheel encoders, 2D Lidar, 3D nodding Lidar, IMU and vision sensors. The abundance of sensing devices and the expandability of the system make it a powerful platform for exploring many different IMM model configurations.

2.2. State Estimation Theory

2.2.1. The Linear Kalman Filter

The Kalman filter is an optimal algorithm for estimating the states of a stochastic linear Gaussian time-invariant system, given the system dynamic model, system inputs and measurement feedback [7]. The algorithm assumes that system processes contain uncertainties and that sensor measurements are corrupted by noise. These uncertainties are modelled as zero-mean Gaussian distributions [6]. The filter operates by first predicting the future states of the system using the process model and inputs. A state covariance matrix is then updated to reflect the variance of the estimated states based on process uncertainties

[6]. The algorithm then uses the current system measurement, measurement estimate and measurement noise to determine the innovation covariance [7]. The innovation covariance is used to determine the correction required to generate an optimal estimate for the system states given the available sensor information [6]. The correction is represented by the Kalman gain matrix, which is determined from the covariance and innovation covariance matrices [7]. This gain matrix is multiplied by the current measurement residual and added to the current uncorrected state estimate vector. The Kalman gain is then used to correct the covariance matrix of the system.

Consider a linear time-invariant system defined by:

$$\dot{\mathbf{x}} = F\mathbf{x} + B\mathbf{u} + G\mathbf{w} \tag{1}$$

where $\mathbf{x}$ is the system state vector, F is the system matrix, B is the input matrix, $\mathbf{u}$ is the input vector, G is the process noise matrix and $\mathbf{w}$ is the process noise vector [38]. The measurement model for this system is:

$$\mathbf{y} = H\mathbf{x} + \boldsymbol{\nu} \tag{2}$$

where $\mathbf{y}$ is the measurement vector, H is the output matrix and $\boldsymbol{\nu}$ is the measurement noise vector [38]. The noise vectors are defined such that $E\langle \mathbf{w}\mathbf{w}^{\mathrm{T}} \rangle = Q_w$ and $E\langle \boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}} \rangle = R_{\nu}$ where $E\langle \cdot \rangle$ denotes the expected value [6]. The linear observer for this system is:

$$\dot{\hat{\mathbf{x}}} = F\hat{\mathbf{x}} + B\mathbf{u} + L(\mathbf{y} - \hat{\mathbf{y}}) \tag{3}$$

where $\hat{\mathbf{x}}$ is the estimated state vector, L is the observer gain and $\hat{\mathbf{y}}$ is the measurement estimate such that $\hat{\mathbf{y}} = H\hat{\mathbf{x}}$ [38]. The linear error state for this system is [38]:

$$\delta\mathbf{x} = \mathbf{x} - \hat{\mathbf{x}} \tag{4}$$

Differentiation of the error state and substitution gives:

$$\delta\dot{\mathbf{x}} = (F - LH)\delta\mathbf{x} + G\mathbf{w} - L\boldsymbol{\nu} \tag{5}$$

The Kalman filter automatically determines the optimal observer gain using the noise parameters for the system. The observer gain L in the observer equations is replaced by the Kalman gain K, given by:

$$K = PH^{\mathrm{T}}R_{\nu}^{-1} \tag{6}$$

where P is the state covariance matrix defined by $E\langle(\mathbf{x} - E\langle\mathbf{x}\rangle)(\mathbf{x} - E\langle\mathbf{x}\rangle)^{\mathrm{T}}\rangle$ [6].

For computer implementations, the Kalman filter is typically used in its discrete form. The system equation for the filter becomes:

$$\hat{\mathbf{x}}_k^- = \Phi\hat{\mathbf{x}}_{k-1}^+ + \Gamma\mathbf{u}_k \tag{7}$$

where Φ is the state transition matrix, Γ is the discrete time input matrix and k is the discrete time increment [6]. Here, the $+$ and $-$ superscripts denote corrected and uncorrected quantities respectively. The state transition matrix is [6]:

$$\Phi = e^{FT} \approx I + FT \tag{8}$$

In this equation, T is the sampling time and I is the identity matrix. The discrete time input matrix is [6]:

$$\Gamma = \int_0^T e^{F(T-\lambda)} B d\lambda \approx BT \tag{9}$$

The discrete time process noise matrix is [6]:

$$Q_d = G_d Q_w G_d^{\mathrm{T}} \tag{10}$$

Where G_d is defined by [6]:

$$G_d = \int_0^T e^{F(T-\lambda)} G d\lambda \approx GT \tag{11}$$

The uncorrected state covariance matrix estimate is [6]:

$$\hat{P}_k^- = \Phi \hat{P}_{k-1}^+ \Phi^{\mathrm{T}} + Q_d \tag{12}$$

and the innovation covariance matrix is given by:

$$S_k = H \hat{P}_k^- H^{\mathrm{T}} + R_d \tag{13}$$

where R_d is the discrete time measurement noise matrix which is equivalent to R_ν [6]. The Kalman gain for the discrete time system is [6]:

$$K_k = H \hat{P}_k^- S_k^{-1} \tag{14}$$

This gain represents the level of trust that the estimator has in the measurement update [7]. The Kalman gain determines how much correction will be applied to the state estimate $\hat{\mathbf{x}}_k^-$ using the measurement residual [7]. The corrected state estimate is given by:

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k(\mathbf{y}_k - H\hat{\mathbf{x}}_k^-) \tag{15}$$

The term $(\mathbf{y}_k - H\hat{\mathbf{x}}_k^-)$ is referred to as innovation [6]. Finally, the state covariance estimate is corrected using [6]:

$$\hat{P}_k^+ = \hat{P}_k^- - K_k H \hat{P}_k^- \tag{16}$$

This process is recursively applied to predict the states of the system for all future time. Generally, accurate initialization of the system states in the prediction model is required for the Kalman filter estimates to converge to the true system states.

The Kalman filter algorithm does have some limitations that need to be considered before implementation. First, the Kalman filter is only optimal if the system dynamics are linear and the system uncertainties are additive Gaussian distributions [7]. For non-linear systems, the Kalman filter may still be applied using modified versions like the EKF, but the solution is no longer optimal [6]. Another limitation of the Kalman filter is that it may require tuning of the noise parameters to effectively track system states [6]. This limitation is further complicated by the selection of the dynamic model which needs to match the true dynamics of the system under study.

2.2.2. **Extended Kalman Filter**

The EKF is a suboptimal version of the Kalman filter that is used for state estimation of nonlinear systems. The algorithm applies the same prediction process as the Kalman Filter but first requires the set of nonlinear dynamic equations to be linearized about a nominal trajectory defined by the most recent state estimate [39].

Considering a nonlinear system model given by:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w}) \tag{17}$$

$$\mathbf{y} = h(\mathbf{x}, \boldsymbol{\nu}) \tag{18}$$

The state observer for this nonlinear system is defined by [6]:

$$\dot{\hat{\mathbf{x}}} = f(\hat{\mathbf{x}}, \mathbf{u}) + K(h(\mathbf{x}, \boldsymbol{\nu}) - h(\hat{\mathbf{x}})) \tag{19}$$

The error state equation, $\delta\mathbf{x} = \mathbf{x} - \hat{\mathbf{x}}$, becomes [6]:

$$\delta\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w}) - f(\hat{\mathbf{x}}, \mathbf{u}) - K(h(\mathbf{x}, \boldsymbol{\nu}) - h(\hat{\mathbf{x}})) \tag{20}$$

The linearization of the error state equation is obtained from a first order Taylor series expansion about the current nominal estimate such that [6]:

$$\begin{aligned} \delta\dot{\mathbf{x}} = f(\hat{\mathbf{x}}, \mathbf{u}) + \left.\frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}} \\ \mathbf{w}=0}} \delta\mathbf{x} + \left.\frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{w}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}} \\ \mathbf{w}=0}} \mathbf{w} - f(\hat{\mathbf{x}}, \mathbf{u}) \\ - K\left(h(\hat{\mathbf{x}}) + \left.\frac{\partial h(\mathbf{x}, \boldsymbol{\nu})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}} \\ \boldsymbol{\nu}=0}} \delta\mathbf{x} + \left.\frac{\partial h(\mathbf{x}, \boldsymbol{\nu})}{\partial \boldsymbol{\nu}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}} \\ \boldsymbol{\nu}=0}} \boldsymbol{\nu} - h(\hat{\mathbf{x}}) \right) \end{aligned} \tag{21}$$

In this case, the nominal estimate is $(\mathbf{x} = \hat{\mathbf{x}}, \mathbf{w} = 0, \boldsymbol{\nu} = 0)$. Simplification of this linearization yields:

$$\delta\dot{\mathbf{x}} = (F - KH)\delta\mathbf{x} + G_w\mathbf{w} - KG_\nu\boldsymbol{\nu} \tag{22}$$

where F is the linearized system matrix, K is the Kalman Gain, H is the linearized output matrix, G_w is the linearized process noise matrix and G_ν is the linearized measurement noise matrix [6]. The filter matrices are summarized below:

$$\begin{aligned} F &= \left.\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{w})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}} \\ \mathbf{w}=0}}, G_w = \left.\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{w})}{\partial \mathbf{w}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}} \\ \mathbf{w}=0}} \\ H &= \left.\frac{\partial h(\mathbf{x},\boldsymbol{\nu})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}} \\ \boldsymbol{\nu}=0}}, G_\nu = \left.\frac{\partial h(\mathbf{x},\boldsymbol{\nu})}{\partial \boldsymbol{\nu}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}} \\ \boldsymbol{\nu}=0}} \end{aligned} \tag{23}$$

Using this linearized error state model, the Kalman filter algorithm as described in section 2.2.1 can be implemented. The nonlinear system equations are used for the uncorrected state estimate step of the Kalman filter, while the linearized filter matrices are used for determining the covariance matrix and Kalman gain.

The EKF has several limitations that can hinder its performance for nonlinear state estimation applications. The filter uses a first order Taylor series expansion to linearize the error dynamics and determine the state covariance. This approximation may not be accurate enough for highly linear systems [9]. If the sampling time is not small enough, the filter can quickly diverge if the states vary significantly between time steps. The linearization process is also computationally expensive, making the filter generally slower than the linear Kalman filter during implementation [6]. This may not be adequate for systems that

are highly nonlinear [10]. Another formulation, the Unscented Kalman filter uses the unscented transform of sample points through a nonlinear function to produce a Gaussian approximation of the function [10]. This can yield more accurate results than the Extended Kalman Filter, however, it can still be computationally expensive [9]. Similarly, optimization techniques for nonlinear systems like Moving-Horizon Estimation (MHE) can outperform an EKF for highly nonlinear systems, however, optimizing the numerical solution for these equations is computationally demanding [40].

2.2.3. **Interactive Multiple Model Filter**

The IMM algorithm is a state prediction method that adaptively predicts the states of systems that have varying dynamics [14]. In general, designing a filter for state estimation requires an accurately defined system model that effectively represents all system dynamics, parameters and inputs. If this information is unavailable, then a filter model must be selected based on the expected behaviour of the system. This can often lead to incorrect assumptions that produce inaccurate predictions, especially if the system dynamics change for different scenarios [14]. For example, an application that can benefit from multiple dynamic models is a mobile robot with caster wheels. The motion characteristics of caster-wheeled robots change frequently when these robots alternate between lateral and longitudinal movements.

The IMM algorithm facilitates the model selection process by running multiple Kalman filters in parallel [19]. Instead of switching between filters for the best state estimate, the IMM estimates are the result of mixing estimates from each filter to yield a

cumulative prediction that is weighted based on the measurement residuals of each model prediction and the measured state [14]. After each measurement, the likelihoods of each model are calculated to determine the contributions of each filter to the mixed state estimate.

The IMM algorithm recursively calculates filter performance and uses conditional probabilities to determine when mode transition is required to maintain an accurate estimation [14]. The formulation here assumes a two-mode filter but can be extended to include any number of modes. The state switching matrix is given by:

$$p^{ij} = \begin{bmatrix} \mathrm{p}_{ii} & \mathrm{p}_{ij} \\ \mathrm{p}_{ji} & \mathrm{p}_{jj} \end{bmatrix} \tag{24}$$

where p_{ij} represents the probability of switching from mode i to j. The elements in the state switching matrix are selected parameters that govern the likelihood of switching modes or remaining in the current mode [14]. The probability of each model is defined by:

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_i & \mu_j \end{bmatrix} \tag{25}$$

The normalization vector for maintaining a total model probability of 1 given N filter modes is calculated as [14]:

$$\bar{\boldsymbol{\Psi}}^j = \sum_{i=1}^{N} p^{ij} \boldsymbol{\mu}^i \tag{26}$$

The conditional model probabilities are used to mix the state estimates and covariance matrices. The conditional probability matrix is given by:

$$\mu^{i|j} = \frac{p^{ij}\hat{\boldsymbol{\mu}}^{i}}{\bar{\boldsymbol{\psi}}^{j}} \tag{27}$$

where $\hat{\boldsymbol{\mu}}^{i}$ is the estimate of the probabilities for each model from the previous time increment [14]. The IMM uses the conditional probabilities and the current state estimate from each individual model to produce a set of mixed state estimates and covariance matrices. The mixed state estimates are calculated by:

$$\hat{\mathbf{X}}^{0j} = \sum_{i=1}^{N} \hat{\mathbf{X}}^{i}\mu^{i|j} \tag{28}$$

where $\hat{\mathbf{X}}^{0j}$ is the mixed state estimate for model j and $\hat{\mathbf{X}}^{i}$ is the current state estimate for model i [14]. For a two-mode system, this equation will yield two mixed states, $\hat{\mathbf{X}}^{01}$ and $\hat{\mathbf{X}}^{02}$, that are a mixture of the state predictions $\hat{\mathbf{X}}^{1}$ and $\hat{\mathbf{X}}^{2}$ given by the individual Kalman filters and their conditional probabilities.

Similarly, the mixed covariance matrix estimates are computed using the current covariance matrix estimate for each individual filter, the state estimates of each filter, the mixed state estimates and the conditional probability matrix [19]. The mixed covariance matrices are given by [14]:

$$\hat{P}^{0j} = \sum_{i=1}^{N} \mu^{i|j}\left[\hat{P}^{i} + \left(\hat{\mathbf{X}}^{i} - \hat{\mathbf{X}}^{0j}\right)\left(\hat{\mathbf{X}}^{i} - \hat{\mathbf{X}}^{0j}\right)^{\mathrm{T}}\right] \tag{29}$$

The innovations and innovation covariance matrices for each model are computed to determine the likelihood of each model. The innovation and innovation covariance matrices are given by:

$$\mathbf{Z}^j = \mathbf{y} - \hat{\mathbf{y}}^j \tag{30}$$

$$S^j = H^j \hat{P}^{0j} (H^j)^{\mathrm{T}} + R \tag{31}$$

where $\mathbf{Z}^j$ is the innovation of model j at the current time increment, $\mathbf{y}$ and $\hat{\mathbf{y}}^j$ are the vectors containing the measurements and measurement estimates of the system states at the current time increment respectively, H^j is the output matrix of model j, S^j is the innovation covariance matrix of model j and R is the covariance matrix of the measurement noise [14]. The likelihoods (Λ^j) of each model matching the current system dynamics are computed by [14]:

$$\Lambda^j = \frac{1}{\sqrt{|2\pi S^j|}} exp\left[-\frac{1}{2}(\mathbf{Z}^j)^{\mathrm{T}}(S^j)^{-1}(\boldsymbol{Z}^j)\right] \tag{32}$$

Using the likelihood of each model, the probability normalizing constant is calculated as [14]:

$$\mathrm{c} = \sum_{i=1}^{N} \Lambda^i \bar{\boldsymbol{\psi}}^i \tag{33}$$

The estimates of the probabilities for each model are updated for the next iteration using [14]:

$$\hat{\boldsymbol{\mu}}^j = \frac{\Lambda^j \bar{\boldsymbol{\psi}}^j}{\mathrm{c}} \tag{34}$$

The final steps involve combining the state estimates with the recently calculated model probabilities to produce an overall system state estimate and system covariance estimate. The combined state estimate is:

$$\hat{\mathbf{X}} = \sum_{i=1}^{N} \hat{\mathbf{X}}^i \hat{\boldsymbol{\mu}}^i \tag{35}$$

and the combined covariance matrix estimate is given by [14]:

$$\hat{P} = \sum_{i=1}^{N} \hat{\boldsymbol{\mu}}^i \left[\hat{P}^i + \left(\hat{\mathbf{X}}^i - \hat{\mathbf{X}}\right)\left(\hat{\mathbf{X}}^i - \hat{\mathbf{X}}\right)^{\mathrm{T}} \right] \tag{36}$$

IMM filters can be designed for nonlinear systems using parallel EKFs. In that case, the linearized filter matrices are used in the IMM filter.

The main disadvantage of using an IMM filter is that it does not give optimal state estimation results [14]. If a system is strictly governed by a set of fixed linear time-invariant dynamic equations, then a Kalman filter derived from that set of equations will yield the optimal estimator solution [7]. Furthermore, it is more computationally efficient to run a single Kalman filter instead of an IMM with multiple modes. The increase in computational complexity of an IMM filter scales with the number of modes that the filter contains. The calculation of this complexity value is not evaluated in this thesis. This is not necessarily a significant issue when using an IMM for nonlinear systems since existing filters like the

EKF are already linearized approximations. An IMM utilizing parallel EKFs can outperform the single model filter if configured and tuned correctly.

Chapter 3

Computer Vision Tracking using the Interactive Multiple Model Filter

About this chapter: This chapter[1] analyses the problem of tracking a target in a video sequence autonomously using a combination of computer vision tracking techniques. The IMM filter is used for target trajectory prediction, and the mean shift algorithm is used for measurement updates.

3.1. Problem Formulation

Computer vision object tracking is a rapidly developing technology that is becoming widely used in many real-world applications. Some of these applications include mobile robot target tracking [40], traffic monitoring [41] and automatic guidance systems [42]. This area of research focuses on finding an object in a video frame and sequentially detecting the same object in successive frames. Some common methods for object tracking include mean shift [43], active contours [44] and Kanade-Lucas-Tomasi (KLT) tracking [45].

[1] This chapter is based on the following publication of the author:
P. J. Glavine, O. D. Silva, G. Mann and R. Gosine, "Color-Based Object Tracking using Mean Shift and Interactive Multiple Model Kalman Filtering," in *Newfoundland Electrical and Computer Engineering Conference (NECEC)*, St. John's, 2017.

Tracking an object in a video sequence requires the target to be defined such that it can be accurately detected between video frames. This process becomes difficult when the video contains object clutter, lighting changes, or a target that changes size, shape or colour for example. Furthermore, the object becomes even more difficult to track when it moves unpredictably in an arbitrary fashion. The object tracking method discussed in this chapter uses a mean shift colour-based approach paired with the IMM filter. The mean shift tracker is used to identify the tracked target using its colour histogram. The position of the target that is calculated by the mean shift tracker is used as the measurement for the IMM filter. The IMM filter estimates the trajectory of the target using a combination of two kinematic motion models, the constant velocity model and constant acceleration model. This implementation allows the tracker to switch between prediction models when the tracked target abruptly changes directions or begins accelerating unexpectedly.

3.2. **Methodology**

3.2.1. **Mean Shift Algorithm**

The mean shift algorithm is a method for finding the mode of a nonparametric dataset through gradient ascension [46]. This is done by iteratively calculating the mean of a set of sample points within a window and shifting a position estimate to the location of the sample data centroid [47]. This method can be applied to computer vision tracking by representing a target using a colour histogram which approximates its probability distribution function (PDF) [48]. The algorithm uses gradient ascension to move an initial

position estimate towards the center of a target candidate in successive images. Convergence occurs when the original target and candidate have matching PDFs [48].

Given an initial position $\mathbf{y}_0$, the mode of a random dataset can be found by iteratively travelling from $\mathbf{y}_0$ to a new location $\mathbf{y}_1$ by a vector defined as [46]:

$$\mathbf{m}_h(\mathbf{y}) = \left[\frac{1}{n_x}\sum_{i=1}^{n_x}\mathbf{x}_i\right] - \mathbf{y}_0 \tag{37}$$

Where $\mathbf{m}_h$ is the mean shift vector, n_x is the number of data points in the current window, and $\mathbf{x}_i$ is the vector containing the x and y coordinates of the i^{th} data point. The position $\mathbf{y}_1$ in Figure 1 defines the centroid, or mean location, of the data points in the circular window.

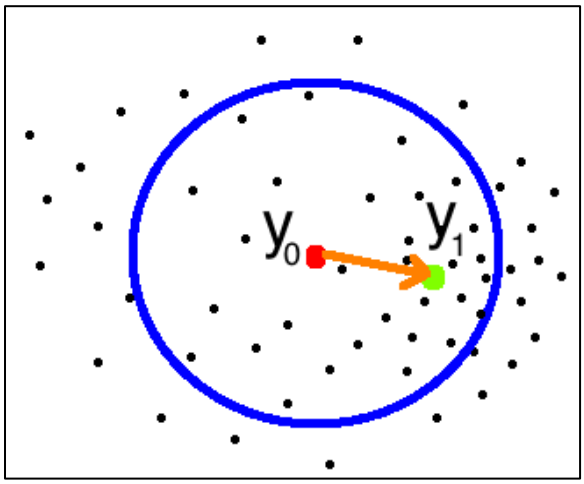


Figure 1 Mean Shift Vector

Weights can be assigned to the data points based on their spatial distance from $\mathbf{y}_0$ by defining a kernel mask for the window [43]. The weighted mean shift vector is given by [46]:

$$\mathbf{m}_h(\mathbf{y}) = \left[\frac{\sum_{i=1}^{n_x} w_i(\mathbf{y}_0)\mathbf{x}_i}{\sum_{i=1}^{n_x} w_i(\mathbf{y}_0)}\right] - \mathbf{y}_0 \tag{38}$$

where w_i is the kernel weight of the i^{th} pixel in the window. The Gaussian kernel has been used in this tracking application, it is defined by [46]:

$$K(\mathbf{x}) = c \cdot exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \tag{39}$$

where c is a constant. This kernel is radially symmetric, therefore it can be expressed as [48]:

$$K(\mathbf{x}) = ck(\|\mathbf{x}\|^{\mathbf{2}}) \tag{40}$$

where k is the kernel profile. Masking a window of a nonparametric data set with a kernel function allows the PDF of the dataset to be approximated as [47]:

$$P(\mathbf{x}) = \frac{1}{n} c \sum_{i=1}^{n} k(\|\mathbf{x} - \mathbf{x}_i\|^2) \tag{41}$$

where $\|\mathbf{x} - \mathbf{x}_i\|^2$ represents the distance from the point $\mathbf{x}$ to the i^{th} data point in the kernel. Higher weights are assigned to points that are closer to $\mathbf{x}$. Differentiating and manipulation of Eq. (41) gives [47]:

$$\nabla P(\mathbf{x}) = \frac{1}{n} c \left[\sum_{i=1}^{n} g_i\right]\left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right] \tag{42}$$

where g_i is the negative gradient of the kernel profile at the i^{th} data point defined such that $g(\mathbf{x}) = -k'(\mathbf{x})$ [48]. The mean shift vector from Eq. (42) is [46]:

$$\mathbf{m}_h(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right] \tag{43}$$

Therefore, the mean shift can be expressed as:

$$\mathbf{m}_h(\mathbf{x}) = \frac{\nabla P(\mathbf{x})}{\frac{1}{n} c \sum_{i=1}^{n} g_i} \tag{44}$$

which shows that the mean shift vector is the gradient of the estimated PDF for the nonparametric dataset [47].

In this vision tracking experiment, the target model is represented using a weighted colour histogram which represents the PDF of the pixels (data points) in the target [48]. The target colour histogram is defined as [43]:

$$\hat{q} = \{\hat{q}_u\}_{u=1..m} \quad \sum_{u=1}^{m} \hat{q}_u = 1 \tag{45}$$

where m is the total number of bins in the colour histogram. The target candidate is given by [43]:

$$\hat{p}(\mathbf{y}) = \{\hat{p}_u(\mathbf{y})\}_{u=1..m} \quad \sum_{u=1}^{m} \hat{p}_u = 1 \tag{46}$$

The target candidate histogram $\hat{p}(\mathbf{y})$ defines a potential match for the original target histogram $\hat{q}$ in the current image frame at location $\mathbf{y}$. The colour histogram of the target candidate can be generated using [43]:

$$\hat{p}_u(\mathbf{y}) = \frac{1}{\sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{\mathrm{h}}\right\|^2\right)} \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right\|^2\right) \delta[b(\mathbf{x}_i) - u] \tag{47}$$

where h is the kernel bandwidth, n_h is the number of pixels within the kernel, $b(\mathbf{x}_i)$ is the colour of the i^{th} pixel in the kernel, u is the set of colour bin values in the range $1..m$ and δ is the Kronecker delta function. The expression $\delta[b(\mathbf{x}_i) - u]$ equals one when the colour of pixel $\mathbf{x}_i$ has the same value as u. When this occurs, the histogram bin u will increase by a normalized value defined by the kernel weight at $\mathbf{x}_i$. Summing this expression over all pixel values in the kernel generates a colour probability distribution [47]. The original target model $\hat{q}$ is calculated using the same method.

The Bhattacharyya coefficient measures the similarity of two probability distributions [49]. It is used to determine if the target candidate matches the original target. The coefficient is defined by [46]:

$$\rho[\hat{p}(\mathbf{y}), \hat{q}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{y})\hat{q}_u} \tag{48}$$

Two probability distributions are similar when the Bhattacharyya coefficient is maximized; this is equivalent to minimizing the distance given by [43]:

$$\mathrm{d} = \sqrt{1 - \rho[\hat{p}(\mathbf{y}), \hat{q}]} \tag{49}$$

Performing a Taylor series expansion of the Bhattacharyya coefficient about an operating point defined as $\hat{\mathbf{y}}_0$ (initial target position) yields [43]:

$$\rho[\hat{p}(\mathbf{y}), \hat{q}] = \rho[\hat{p}(\hat{\mathbf{y}}_0), \hat{q}] + \frac{C_h}{2} \sum_{u=}^{m} \left[\sum_{u=1}^{m} \delta[b(\mathbf{x}_i) - u] \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}} \right] k\left(\left\| \frac{\mathbf{y} - \mathbf{x}_i}{\mathrm{h}} \right\|^2 \right) \tag{50}$$

where C_h is a normalization constant. This linear approximation assumes that $\hat{p}(\mathbf{y})$ does not change significantly from the initial estimate $\hat{p}(\hat{\mathbf{y}}_0)$ when the distance function d is minimized for the current frame [43]. This is a reasonable assumption when the target does not move substantial distances between frames. Maximizing the Bhattacharyya coefficient is dependent on the maximization of the second term in Eq. (50). This equation contains the weights which are given by [47]:

$$w_i(\mathbf{y}_0) = \sum_{u=1}^{m} \delta[b(\mathbf{x}_i) - u] \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}} \tag{51}$$

Combining the weights from Eq. (51) and the gradient form of the mean shift vector in Eq. (43) gives [43]:

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n} \mathbf{x}_i w_i g\left(\left\| \frac{\mathbf{y}_0 - \mathbf{x}_i}{\mathrm{h}} \right\|^2 \right)}{\sum_{i=1}^{n} w_i \, g\left(\left\| \frac{\mathbf{y}_0 - \mathbf{x}_i}{\mathrm{h}} \right\|^2 \right)} \tag{52}$$

The initialized position $\hat{\mathbf{y}}_0$ is iteratively updated using the new position $\hat{\mathbf{y}}_1$ until the distance between the distributions $\hat{p}(\mathbf{y})$ and $\hat{q}$ is minimized below a selected threshold [43].

3.2.2. **Interactive Multiple Model Filter Tracking Implementation**

For this application, the IMM filter uses two linear system models to predict the kinematic states of a "blob" in two-dimensional space. The two kinematic models that were used for tracking targets in this system are the constant velocity (CV) and the constant acceleration (CA) models.

The CV model assumes that the target has a constant velocity, with acceleration considered to be a random walk process of zero mean Gaussian noise [14]. The system dynamics are given by:

$$\begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ \mathbf{a}_x \\ \mathbf{a}_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{v}_x \\ \mathbf{v}_y \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}_{ax} \\ \boldsymbol{\eta}_{ay} \end{bmatrix} \tag{53}$$

where $(\mathbf{x}, \mathbf{y}), (\mathbf{v}_x, \mathbf{v}_y)$ and $(\mathbf{a}_x, \mathbf{a}_y)$ are pixel coordinates, velocities and accelerations respectively. The noise vector $[\boldsymbol{\eta}_{ax}\ \boldsymbol{\eta}_{ay}]^T$ represents the random walk acceleration process [14].

The CA model assumes that the target has a constant acceleration, with variation in acceleration (jerk) modelled as a random walk process of zero mean Gaussian noise [14]. The system dynamics are given by:

$$\begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ \mathbf{a}_x \\ \mathbf{a}_y \\ \dot{\mathbf{a}}_x \\ \dot{\mathbf{a}}_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{v}_x \\ \mathbf{v}_y \\ \mathbf{a}_x \\ \mathbf{a}_y \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}_{jx} \\ \boldsymbol{\eta}_{jy} \end{bmatrix} \tag{54}$$

Here, $(\dot{\mathbf{a}}_x, \dot{\mathbf{a}}_y)$ represent the jerk that the blob undergoes defined by the random noise vector $\left[\boldsymbol{\eta}_{jx}\ \boldsymbol{\eta}_{jx}\right]^T$ [14].

In both cases, the process and measurement noise covariance matrices are given by:

$$Q = \begin{bmatrix} {\sigma_{Qx}}^2 & 0 \\ 0 & {\sigma_{Qy}}^2 \end{bmatrix} \quad R = \begin{bmatrix} {\sigma_{Rx}}^2 & 0 \\ 0 & {\sigma_{Ry}}^2 \end{bmatrix} \tag{55}$$

where ${\sigma_Q}^2$ and ${\sigma_R}^2$ are the variances of the process and measurement noises respectively [6]. The process and measurement noise variances are tuned to match the uncertainty in system motion and position measurements respectively.

3.2.3. **Vision Tracker Design**

The tracking algorithm was implemented as shown in Figure 2 using MATLAB. The mean shift begins with the initialization of a target model by the user. A Gaussian filter is applied to generate a set of weights for the target model. An indexed colour map is obtained from the target model and is used to generate the colour histogram. In the next frame, a target candidate window is initialized from the target coordinates in the previous frame. The new target candidate colour histogram is generated and the weighted mean shift vector is computed. The target position is updated, the weights are recomputed and the Bhattacharyya distance is calculated using the current target candidate and the target model. The mean shift process repeats until the Bhattacharyya distance is below the convergence threshold value.

The Kalman filter models use the mean shift position estimate as a measurement to update the system states. The estimates and covariance matrices from each Kalman filter are inputs for the IMM filter which computes the combined state estimate. Using the available measurements, the probabilities of each filter model are calculated and the combined weighted estimate from each filter is computed to yield the overall IMM estimate.

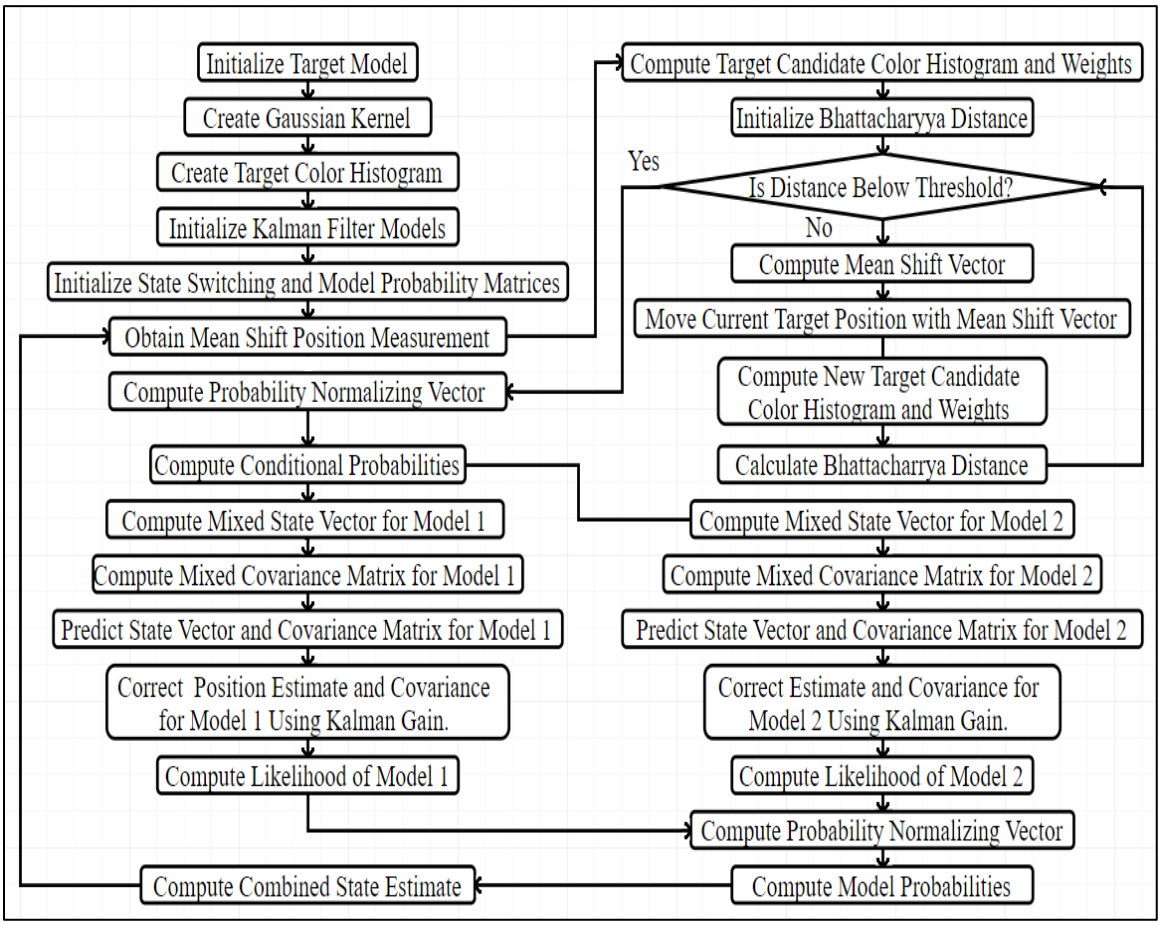


Figure 2 Mean Shift IMM Tracker Algorithm

3.3. Computer Vision Tracker Experiments

3.3.1. Constant Velocity Tracking

The first test involved tracking a red circular target moving in a horizontal straight line with a constant velocity. The center of the circle contains a blue pixel that was used for calculating position tracking error. The constant velocity test is shown in Figure 3 below.

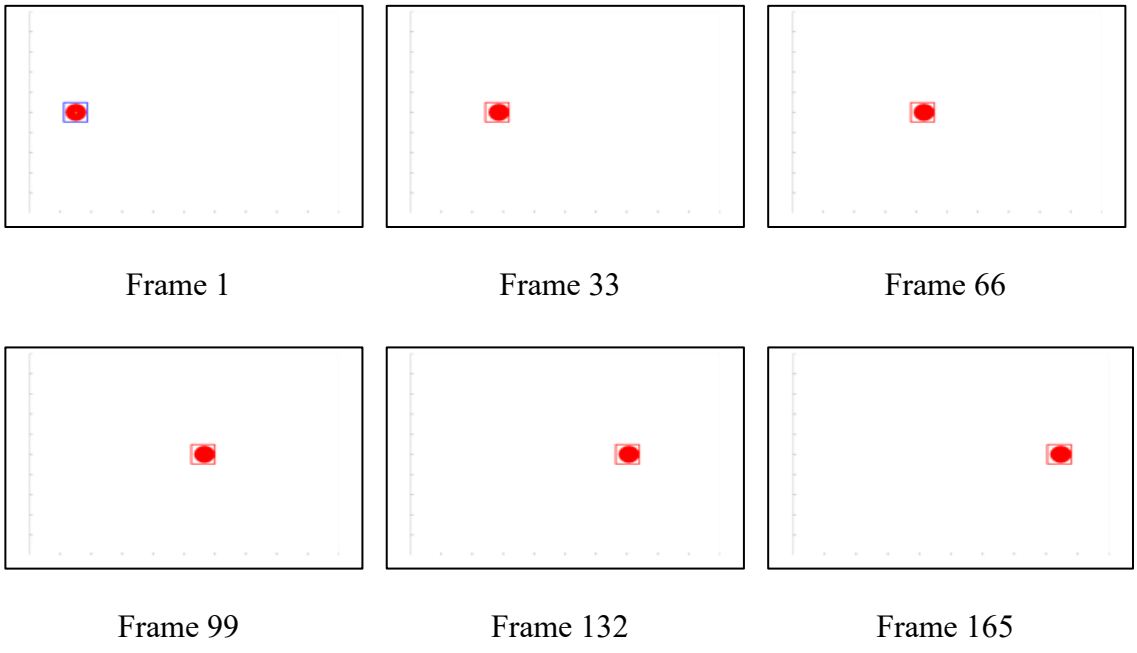


Figure 3 Constant Velocity Test

The tracking results are shown in Table 1. Kalman and IMM filter results are based on five test trial averages since the target motion and measurements are considered Gaussian processes. The addition of Kalman filtering improves tracking accuracy in all cases. The IMM filter outperforms both single-filter models.

Table 1 Constant Velocity Tracking Results

Mean Shift Tracking										
Mean Error X (Pixels)			Mean Error Y (Pixels)				Frames per Measurement			
4.4121			0				1			
Mean Shift with Constant Velocity Kalman Filter Model (5 Trial Average)										
Mean Error X (Pixels)		Mean Error Y (Pixels)		$\sigma_{Qx}{}^2$		$\sigma_{Qy}{}^2$		$\sigma_{Rx}{}^2$	$\sigma_{Ry}{}^2$	Frames per Measurement
2.5054		0		0.01		0.01		40	0	1
Mean Shift and Constant Acceleration Kalman Filter Model (5 Trial Average)										
Mean Error X (Pixels)		Mean Error Y (Pixels)		$\sigma_{Qx}{}^2$		$\sigma_{Qy}{}^2$		$\sigma_{Rx}{}^2$	$\sigma_{Ry}{}^2$	Frames per Measurement
2.2634		0		3		3		20	0	1
Mean Shift and IMM Kalman Filter Model (5 Trial Average)										
Mean Error X (Pixels)	Mean Error Y (Pixels)	$\sigma_{Qx}{}^2$ (CV)	$\sigma_{Qy}{}^2$ (CV)	$\sigma_{Qx}{}^2$ (CA)	$\sigma_{Qy}{}^2$ (CA)	$\sigma_{Rx}{}^2$ (CV)	$\sigma_{Ry}{}^2$ (CV)	$\sigma_{Rx}{}^2$ (CA)	$\sigma_{Ry}{}^2$ (CA)	Frames per Measurement
1.3697	0	0.01	0.01	3	3	40	0	20	0	1

The model probabilities during the constant velocity test are shown in Figure 4. Initially, the constant acceleration model obtains a higher probability. This is because the target instantaneously transitions from a resting state to a constant velocity motion at the beginning of the video sequence. The IMM filter velocities and accelerations are initialized with values of zero, therefore, the filter initially lags behind the motion of the target. The filter estimates a transient period of target acceleration before the system reaches a steady state. Once this occurs, the acceleration becomes a low value, and the probabilities of the constant velocity and acceleration models rise and lower respectively.

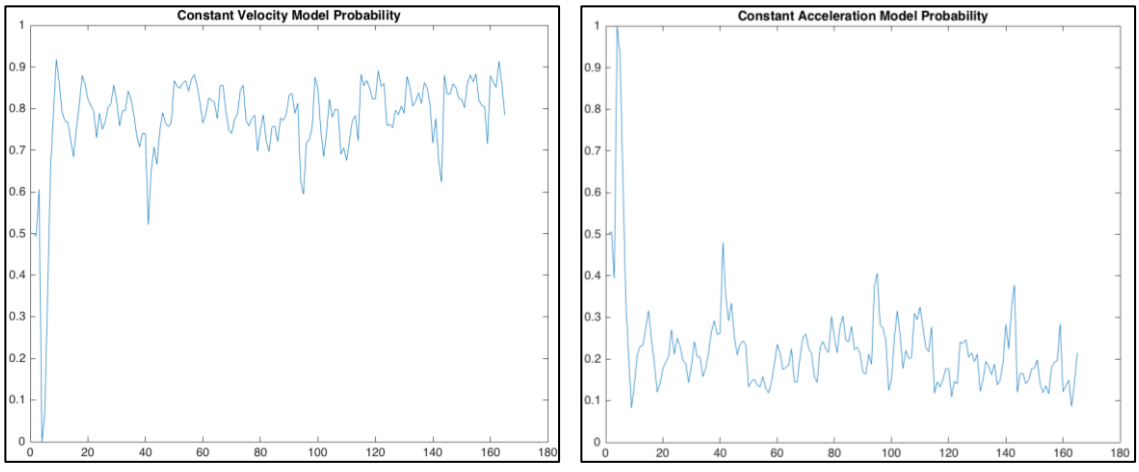


Figure 4 Model Probabilities Constant Velocity Test

3.3.2. **Constant Acceleration Tracking**

The second test involved tracking a red circular target with a blue pixel center moving in a horizontal straight line with a constant acceleration. The actual center position for all frames was calculated by sampling the blue pixel location for ten frames and taking an average to determine the acceleration. The constant acceleration test is shown in Figure 5.

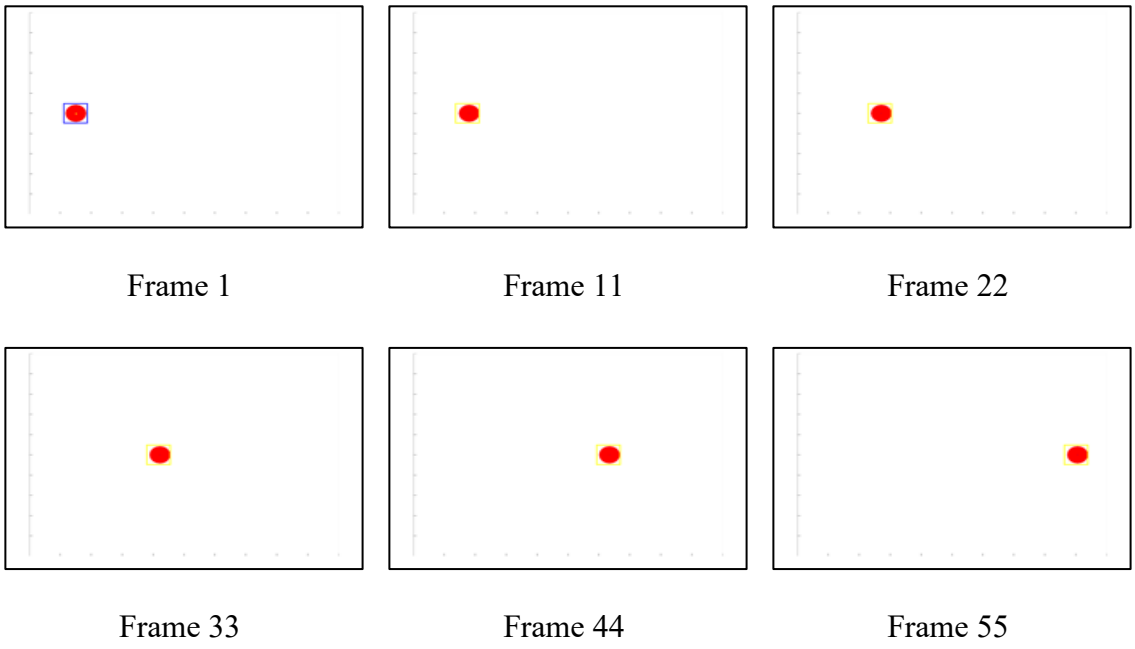


Figure 5 Constant Acceleration Test

As shown in Table 2, Kalman and IMM filtering improve tracking accuracy in all cases over using the mean shift method alone. Again, the IMM filter outperforms both single-filter models, however, not as substantially in this case. For this test, the single filter CA model also accurately tracks the target for all frames because there is no transient acceleration period at the beginning of the video sequence like in the CV test.

Table 2 Constant Acceleration Tracking Results

Mean Shift Tracking										
Mean Error X (Pixels)		Mean Error Y (Pixels)			Frames per Measurement					
5.9416		0			1					
Mean Shift with Constant Velocity Kalman Filter Model (5 Trial Average)										
Mean Error X (Pixels)		Mean Error Y (Pixels)		$\sigma_{Qx}{}^2$	$\sigma_{Qy}{}^2$	$\sigma_{Rx}{}^2$	$\sigma_{Ry}{}^2$	Frames per Measurement		
3.806		0		3	3	50	0	1		
Mean Shift and Constant Acceleration Kalman Filter Model (5 Trial Average)										
Mean Error X (Pixels)		Mean Error Y (Pixels)		$\sigma_{Qx}{}^2$	$\sigma_{Qy}{}^2$	$\sigma_{Rx}{}^2$	$\sigma_{Ry}{}^2$	Frames per Measurement		
2.9628		0		0.0001	0.0001	50	0	1		
Mean Shift and IMM Kalman Filter Model (5 Trial Average)										
Mean Error X (Pixels)	Mean Error Y (Pixels)	$\sigma_{Qx}{}^2$ (CV)	$\sigma_{Qy}{}^2$ (CV)	$\sigma_{Qx}{}^2$ (CA)	$\sigma_{Qy}{}^2$ (CA)	$\sigma_{Rx}{}^2$ (CV)	$\sigma_{Ry}{}^2$ (CV)	$\sigma_{Rx}{}^2$ (CA)	$\sigma_{Ry}{}^2$ (CA)	Frames per Measurement
2.9309	0	3	3	0.001	0.001	50	0	50	0	1

3.3.3. **Elliptic Path Tracking**

The next testing setup for the tracking systems was comprised of a red circular target moving in a circular path with a constant angular velocity and radius relative to the center point. The sequence was generated by plotting markers on a figure in MATLAB and using the "getframe" function to build a video. The center of the red circle contains a blue pixel which was used to accurately determine the target center for all frames. The aspect ratio of the MATLAB figure caused the circular motion to become slightly elliptic in the video sequence, therefore, the motion contained small tangential acceleration components at different points along the path. The actual center position for all frames was calculated by measuring the blue pixel location when the target angle with respect to the center was 0^0, 90^0, 180^0 and 270^0. These pixel locations were used to calculate the major and minor axes of the ellipse and a set of actual target center locations was estimated using:

$$x = x_c + \mathrm{a}\cos\theta \quad y = y_c + \mathrm{b}\sin\theta \tag{56}$$

where x and y are the target coordinates, x_c and y_c are the ellipse center coordinates, a is the major axis, b is the minor axis and θ is the angle to point (x, y) measured from the horizontal axis at the ellipse center. The elliptic path test is shown in Figure 6.

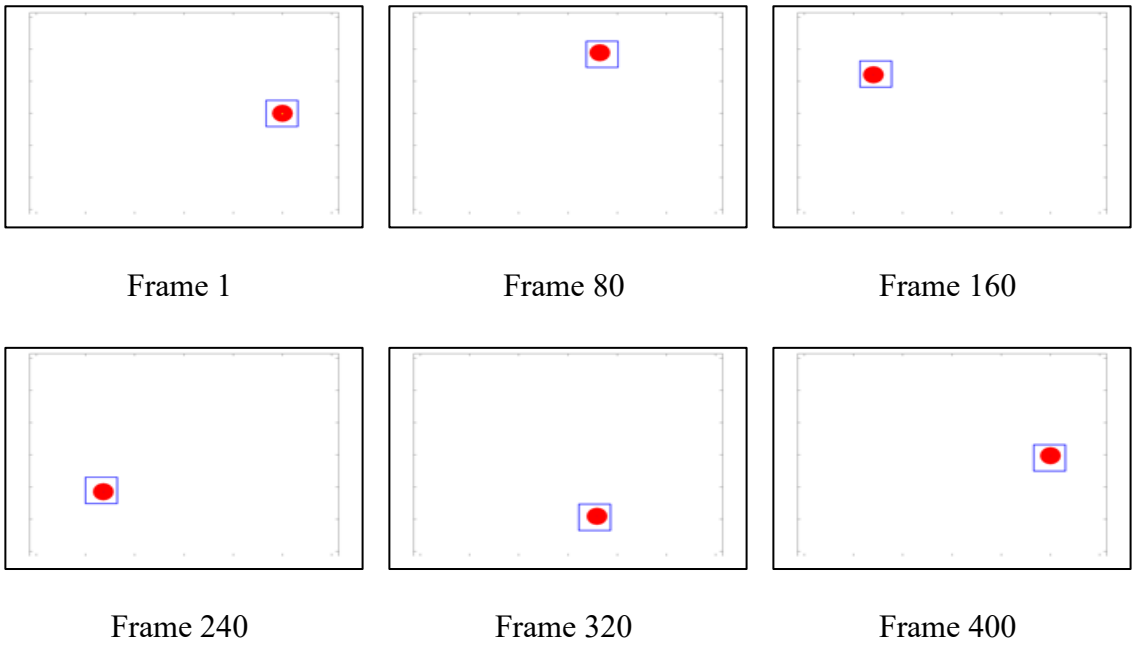


Figure 6 Elliptic Path Test

Table 3 Elliptic Path Tracking Results

Mean Shift Tracking										
Mean Error X (Pixels)		Mean Error Y (Pixels)			Frames per Measurement					
5.4363		4.9410			1					
Mean Shift with Constant Velocity Kalman Filter Model (5 Trial Average)										
Mean Error X (Pixels)		Mean Error Y (Pixels)		σ_{Qx}^2	σ_{Qy}^2	σ_{Rx}^2	σ_{Ry}^2	Frames per Measurement		
5.4373		4.9313		25	5	0.2	0.3	1		
Mean Shift and Constant Acceleration Kalman Filter Model (5 Trial Average)										
Mean Error X (Pixels)		Mean Error Y (Pixels)		σ_{Qx}^2	σ_{Qy}^2	σ_{Rx}^2	σ_{Ry}^2	Frames per Measurement		
5.4538		4.9006		20	10	0.3	0.5	1		
Mean Shift and IMM Kalman Filter Model (5 Trial Average)										
Mean Error X (Pixels)	Mean Error Y (Pixels)	σ_{Qx}^2 (CV)	σ_{Qy}^2 (CV)	σ_{Qx}^2 (CA)	σ_{Qy}^2 (CA)	σ_{Rx}^2 (CV)	σ_{Ry}^2 (CV)	σ_{Rx}^2 (CA)	σ_{Ry}^2 (CA)	Frames per Measurement
5.4330	4.9322	25	8	20	8	0.3	0.3	0.3	0.3	1

Elliptic path tracking results are found in Table 3. All results involving a Kalman or IMM filter are based on averages from five test trials. The results from this test are

mixed, the single Kalman filter models both improved the tracking results in the Y direction over using the mean shift tracker alone, however, both filters performed slightly worse than the standalone mean shift for X direction tracking. The IMM filter slightly improved tracking accuracy in both directions but failed to improve Y direction tracking as much as the single Kalman filter models.

3.3.4. **Constant Acceleration with Occlusion Test**

The constant acceleration occlusion test includes an additional circular target travelling perpendicular to the path of the red circle. The green circle intersects the path of the red circle during the middle frame of the sequence, completely covering the target. The constant acceleration occlusion test is shown in Figure 7.

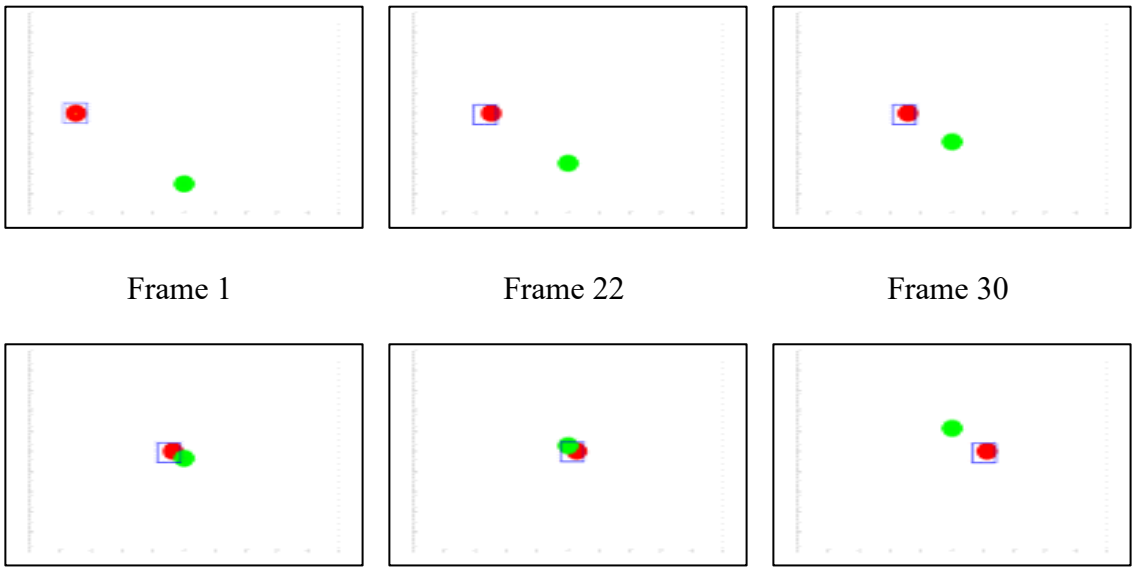


Figure 7 Constant Acceleration with Occlusion Test

Table 4 Constant Acceleration with Occlusion Test

Mean Shift Tracking						
Tracking Success Rate (%)		Frames per Measurement			Convergence Max Iterations	
0		1			20	
Mean Shift with Constant Velocity Kalman Filter Model (5 Trial Average)						
Tracking Success Rate (%)	$\sigma_{Qx}{}^2$	$\sigma_{Qy}{}^2$	$\sigma_{Rx}{}^2$	$\sigma_{Ry}{}^2$	Frames per Measurement	Convergence Max Iterations
100	0.001	0.001	2	2	3	10
Mean Shift and Constant Acceleration Kalman Filter Model (5 Trial Average)						
Tracking Success Rate (%)	$\sigma_{Qx}{}^2$	$\sigma_{Qy}{}^2$	$\sigma_{Rx}{}^2$	$\sigma_{Ry}{}^2$	Frames per Measurement	Convergence Max Iterations
100	0.001	0.001	2	2	3	10

Mean Shift and IMM Kalman Filter Model (5 Trial Average)										
Tracking Success Rate (%)	$\sigma_{Qx}{}^2$ (CV)	$\sigma_{Qy}{}^2$ (CV)	$\sigma_{Qx}{}^2$ (CA)	$\sigma_{Qy}{}^2$ (CA)	$\sigma_{Rx}{}^2$ (CV)	$\sigma_{Ry}{}^2$ (CV)	$\sigma_{Rx}{}^2$ (CA)	$\sigma_{Ry}{}^2$ (CA)	Frames per Measurement	Convergence Max Iterations
100	0.1	0.001	0.1	0.001	40	0.1	40	0.1	3	10

The test results are shown in Table 4. The mean shift algorithm was unable to track the target while it is briefly occluded by the green circle. The number of frames per measurement was increased for the Kalman and IMM filter tests to demonstrate their ability to track without mean shift measurements for short periods of time. The filters successfully tracked the target for all trials tested. The Kalman and IMM filters use the object kinematic state estimates to continue tracking the target when mean shift measurements are unavailable or unreliable.

3.3.5. **General Tracking Results**

The designed vision tracker is capable of tracking objects in real-world video sequences as shown in Figure 8. The red ball is accurately tracked for the entire duration of the video including instances where the ball is partially occluded by the juggler's hand. The blue box indicates the tracking window and the green dotted path is the trajectory of the ball. The tracking demonstration in Figure 8 is available for viewing online at https://www.youtube.com/watch?v=O4t1poYL6rw.

Figure 8 IMM-Mean Shift Juggling Tracking [50]

3.4. **Conclusions**

The IMM filter improved the tracking results of the mean shift method for all cases. Blending the predictions of different kinodynamic models together allows the tracker to react quickly to abrupt changes in the motion characteristics of the blob. The results of the CV test indicate that the IMM filter is more reliable than a single Kalman filter model when instantaneous kinematic transitions occur in the behaviour of the target. Using the IMM filter did not yield a large difference in tracking accuracy for the synthetic videos involving the red circle. This is because the tested motion paths were relatively simple and stable,

therefore, the tracking estimation of each evaluated technique was quite accurate. It is expected that for target tracking in unstructured scenarios where the object motion characteristics are dynamic and change rapidly, the added computational cost of using an IMM filter may be worth the improved tracking accuracy. The IMM filter kinematic state estimates also allow the system to effectively follow targets that are briefly occluded by objects which is a frequent problem in most real-world video tracking applications.

Another observation that was qualitatively analyzed during the tracker testing concerns the computational improvements of pairing the mean shift algorithm with a Kalman or IMM filter. The mean shift algorithm runs multiple, intensive image processing operations in several loops before PDF similarity convergence occurs. The IMM filter only needs to perform several matrix operations to update the system states. Since the IMM filter can accurately track a target for multiple frames without a mean shift measurement update, it can reduce the frequency of mean shift computations and improve algorithm efficiency. The extent of this efficiency improvement is a function of mean shift convergence threshold, video resolution, frames per measurement and several other factors. The quantitative analysis of this observation is not covered in this thesis but can be examined in future work with the vision tracker.

Chapter 4

Vehicle Inertial Navigation System using the Interactive Multiple Model Filter

About this chapter: This chapter[2] analyses an INS state estimation system for an automobile. The system uses a vehicle kinematic model to predict motion and corrects the estimates using GPS and heading sensor feedback. The state estimator uses an IMM filter that uses differently tuned noise parameters to improve estimator performance. The varied noise parameters allow the filter to shift its confidence between the GPS and heading sensors when one sensor more correctly reflects the actual trajectory of the vehicle.

4.1. Problem Formulation

Localization is one of the first major tasks required for a fully autonomous system to function properly [7]. This is a diverse problem which is necessary for accurate tracking of robot or vehicle movement during operations. Without accurate localization, a system cannot be controlled safely or perform tasks with precision. Furthermore, if an autonomous

[2] This chapter is based on the following publication of the author:
P. J. Glavine, O. D. Silva, G. Mann and R. Gosine, "GPS Integrated Inertial Navigation System Using Interactive Multiple Model Extended Kalman Filtering," in *2018 Moratuwa Engineering Research Conference (MERCon)*, Moratuwa, 2018.

system has rapidly changing dynamics, it can be difficult to maintain accurate localization estimates using a single process or measurement model [19].

To address this problem, an INS has been designed using an IMM framework. The vehicle kinematics are predicted using a model that treats the vehicle as a three-dimensional frame in space that can rotate about three axes. The IMM model uses two sets of tuned noise parameters that allow the system to vary its confidence in the available feedback sensors. This can improve system performance in situations where a sensor becomes less reliable for predicting system states, especially during abrupt manoeuvres. The filter, in this case, allows the system to continuously switch between sets of noise parameters, as needed, to better track the actual trajectory of the vehicle.

4.2. **Methodology**

4.2.1. **KITTI Vision Benchmark Data Set**

The vehicle in this study is a 6 degree of freedom system that is equipped with an OXTS RT3003 sensor that includes a built-in IMU and DGPS unit that operates using the World Geodetic System 84 (WGS84) model [22]. The data set is obtained from a Volkswagen Passat B6 driving through a residential area in Karlsruhe, Germany [22]. The vehicle setup is shown in Figure 9. The IMU provides body frame acceleration and angular velocity measurements; the DGPS unit gives accurate positional readings that are used as the ground truth coordinates in this study. The body frame coordinates of the INS are the same as the GPS/IMU frame.

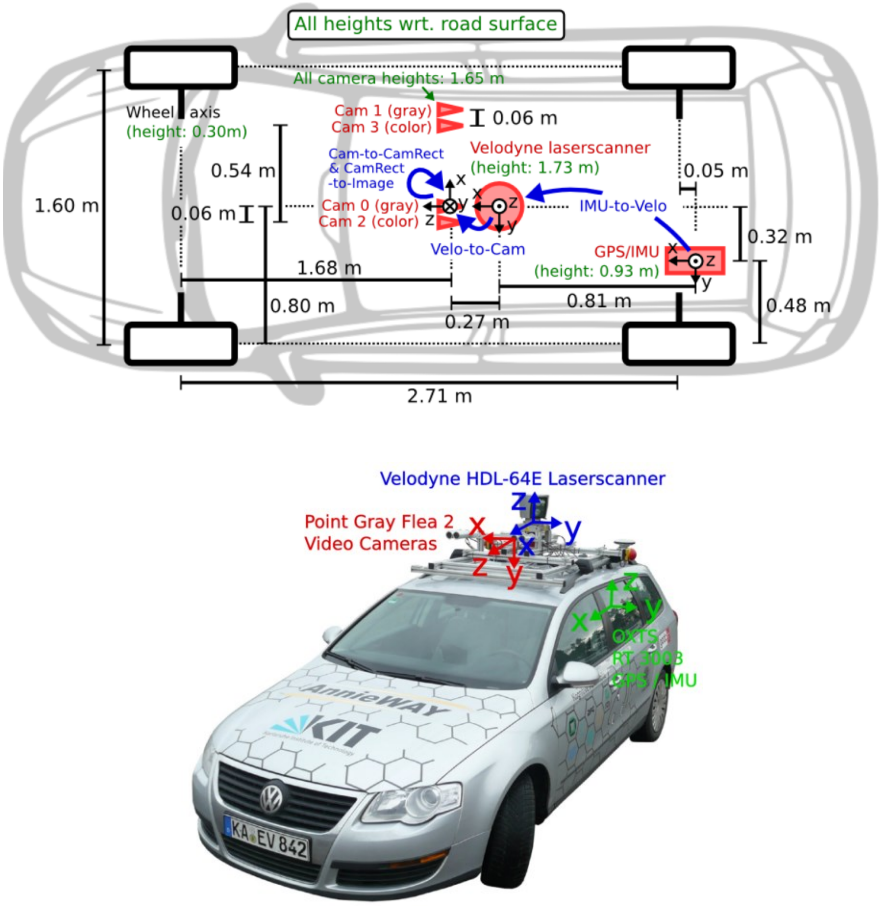


Figure 9 Vehicle and Sensor Configuration [22]

4.2.2. **Vehicle State Space Model**

The vehicle state space model is a nonlinear system. The model considers the vehicle to be a moving frame in three-dimensional space that can rotate about three axes. Changes in the states of this system are a function of states, control inputs and process noise. The state space model is:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w}) \tag{57}$$

where $\mathbf{x}$ is the state vector, $\mathbf{u}$ is the system input vector and $\mathbf{w}$ is the process noise vector [6]. The measurement model for the system is also a nonlinear function which is given by:

$$\mathbf{y} = h(\mathbf{x}, \mathbf{v}) \tag{58}$$

where $\mathbf{y}$ is the measurement vector and $\boldsymbol{\nu}$ is the measurement noise vector [6].

The estimated states of the system are included in the following state vector:

$$\mathbf{x} = [\mathbf{p}, \mathbf{v}, \mathbf{q}, \mathbf{b}_a, \mathbf{b}_\omega]^T \tag{59}$$

where $\mathbf{p} = [p_x, p_y, p_z]^T$ are the vehicle position coordinates with respect to the world frame represented in the world frame, $\mathbf{v} = [v_x, v_y, v_z]^T$ are the vehicle velocities with respect to the world frame expressed in the body frame, $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$ is the quaternion that rotates a vector from the body frame to the world frame, $\mathbf{b}_a = [b_{ax}, b_{ay}, b_{az}]^T$ represents the accelerometer bias vector in the body frame and $\mathbf{b}_\omega = [b_{\omega x}, b_{\omega y}, b_{\omega z}]^T$ is the gyroscope bias vector in the body frame [51]. The choice of representing rotations with quaternion vectors is explained shortly.

The inputs for the state space model are given by an IMU. The IMU measurements are integrated over discrete time steps using the system kinematic equations to generate an estimated trajectory of the system states. The input vector is defined by:

$$\mathbf{u} = [\mathbf{f}_m, \boldsymbol{\omega}_m]^T \tag{60}$$

where $\mathbf{f}_{\mathrm{m}}$ is the measured body frame linear acceleration and $\boldsymbol{\omega}_{\mathrm{m}}$ is the measured body frame angular velocity provided by the IMU accelerometer and gyroscope respectively. The model for the IMU accelerometer is:

$$\mathbf{f_m} = \mathbf{a} + \mathbf{b_a} - R_{\mathrm{q}}^{\mathrm{T}}\mathbf{g_e} + \boldsymbol{\eta}_{\mathbf{fm}} \tag{61}$$

where $\mathbf{a} = [\mathrm{a_x}, \mathrm{a_y}, \mathrm{a_z}]^{\mathrm{T}}$ is the acceleration vector of the body frame with respect to the world frame represented in the body frame, R_{q} is the rotation matrix that rotates a vector from the body frame to the world frame, $\mathbf{g_e} = [0\ 0\ 9.81]^{\mathrm{T}}$ is the Earth's gravity vector represented in the world frame in $\mathrm{m/s^2}$, $\boldsymbol{\eta}_{\mathrm{fm}}$ is zero-mean Gaussian noise such that $\boldsymbol{\eta}_{\mathrm{fm}} \sim N(0, \boldsymbol{\sigma}_{\mathrm{fm}}^2)$ and $\boldsymbol{\sigma}_{\mathrm{fm}}^2$ is the accelerometer noise variance [51]. The gravity vector is transformed into the vehicle body frame coordinates and subtracted from the measured accelerometer reading. Variations in the accelerometer bias are modelled as a random walk process such that:

$$\dot{\mathbf{b}}_{\mathrm{a}} = \boldsymbol{\eta}_{\mathrm{ba}} \tag{62}$$

where $\boldsymbol{\eta}_{\mathrm{ba}}$ is zero-mean Gaussian noise such that $\boldsymbol{\eta}_{\mathrm{ba}} \sim N(0, \boldsymbol{\sigma}_{\mathrm{ba}}^2)$ and $\boldsymbol{\sigma}_{\mathrm{ba}}^2$ is the bias noise variance [6]. Rearranging the accelerometer measurement equation and isolating the vehicle acceleration vector gives:

$$\mathbf{a} = \mathbf{f}_{\mathrm{m}} - \mathbf{b}_{\mathrm{a}} + R_{\mathbf{q}}^{\mathrm{T}}\mathbf{g_e} - \boldsymbol{\eta}_{\mathrm{fm}} \tag{63}$$

The model for the IMU gyroscope sensor is:

$$\boldsymbol{\omega}_{\mathrm{m}} = \boldsymbol{\omega} + \mathbf{b}_{\omega} + \boldsymbol{\eta}_{\omega\mathrm{m}} \tag{64}$$

where $\boldsymbol{\omega} = [\omega_{\mathrm{x}}, \omega_{\mathrm{y}}, \omega_{\mathrm{z}}]^{\mathrm{T}}$ is the angular velocity vector of the body frame with respect to the world frame represented in the body frame and $\boldsymbol{\eta}_{\omega\mathrm{m}}$ is zero-mean Gaussian noise such that $\boldsymbol{\eta}_{\omega\mathrm{m}} \sim N(0, \boldsymbol{\sigma}^2_{\omega\mathrm{m}})$ and $\boldsymbol{\sigma}^2_{\omega\mathrm{m}}$ is the gyroscope noise variance [51]. Variations in the gyroscope bias are also modelled as a random walk process such that:

$$\dot{\mathbf{b}}_{\omega} = \boldsymbol{\eta}_{\mathrm{b}\omega} \tag{65}$$

where $\boldsymbol{\eta}_{\mathrm{b}\omega}$ is zero-mean Gaussian noise such that $\boldsymbol{\eta}_{\mathrm{b}\omega} \sim N(0, \boldsymbol{\sigma}^2_{\mathrm{b}\omega})$ and $\boldsymbol{\sigma}^2_{\mathrm{b}\omega}$ is the bias noise variance [6]. Gathering the noise terms for the accelerometer and gyroscope yields the process noise vector:

$$\mathbf{w} = [\boldsymbol{\eta}_{\mathrm{fm}}, \boldsymbol{\eta}_{\omega\mathrm{m}}, \boldsymbol{\eta}_{\mathrm{ba}}, \boldsymbol{\eta}_{\mathrm{b}\omega}]^{\mathrm{T}} \tag{66}$$

The Euler angle representation for rotations can cause numerical singularities when a system performs certain rotation transitions [6]. To avoid this problem, the quaternion approach has been selected. For this application, rotations are represented by a unit quaternion vector that has the normality property $\|\mathbf{q}\| = 1$ [52]. Quaternions are represented as generalized complex numbers with four components such that $\mathbf{q} = \mathrm{q}_0 + \mathrm{q}_1\mathbf{i} + \mathrm{q}_2\mathbf{j} + \mathrm{q}_3\mathbf{k}$ [53]. A three-dimensional rotation of a vector using quaternions can be represented as a single rotation by an angle θ about an axis $\hat{\mathbf{n}}$ such that [53]:

$$\mathbf{q} = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\hat{\mathbf{n}} \tag{67}$$

The conjugate of a quaternion is given as [53]:

$$\mathbf{q}^* = [q_0, -q_1, -q_2, -q_3]^T \tag{68}$$

Considering an arbitrary vector $\boldsymbol{\rho}$ in $\mathbb{R}^3$ space; this vector can be represented in the quaternion form as $\overline{\boldsymbol{\rho}} = (0\ \boldsymbol{\rho}^T)^T$ [6]. The rotation of this augmented vector can be determined by:

$$\overline{\boldsymbol{\rho}}' = \mathbf{q} \otimes \overline{\boldsymbol{\rho}} \otimes \mathbf{q}^* \tag{69}$$

where $\overline{\boldsymbol{\rho}}'$ is the vector $\overline{\boldsymbol{\rho}}$ rotated by an angle θ about the axis $\hat{\mathbf{n}}$ [52]. In the above context, the $\otimes$ operator represents quaternion multiplication. The product of two quaternions $\mathbf{q}$ and $\mathbf{p}$ is [52]:

$$\begin{aligned}\mathbf{q} \otimes \mathbf{p} = (q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3) + (q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2)\mathbf{i} \\ +(q_0p_2 - q_1p_3 + q_3p_1 + q_2p_0)\mathbf{j} + (q_0p_3 + q_1p_3 - q_2p_1 + q_3p_0)\mathbf{k}\end{aligned} \tag{70}$$

Modifying the quaternion representation into a vector form yields:

$$\overline{\mathbf{q}} = q_0 + \vec{\mathbf{q}} \tag{71}$$

where $\vec{\mathbf{q}} = [q_1, q_2, q_3]^T$ [6]. Using this form of the quaternion, the matrix equivalents for quaternion multiplication can be defined. The left and right quaternion-product matrices are [52]:

$$Q^+ = \begin{bmatrix} q_0 & -\vec{\mathbf{q}}^T \\ \vec{\mathbf{q}} & (q_0 I + [\vec{\mathbf{q}}\times]) \end{bmatrix} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \tag{72}$$

$$Q^{-} = \begin{bmatrix} q_0 & -\vec{\mathbf{q}}^{\mathrm{T}} \\ \vec{\mathbf{q}} & (q_0 I - [\vec{\mathbf{q}}\times]) \end{bmatrix} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \tag{73}$$

where I is the identity matrix and $[\vec{\mathbf{q}}\times]$ is the skew-symmetric form of the vector $\vec{\mathbf{q}}$. This form is written as [52]:

$$[\vec{\mathbf{q}}\times] = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{74}$$

With the quaternion multiplication matrices defined, the rotation of a vector between frames can be expressed in a more compact form. The vector rotation becomes [6]:

$$\overline{\boldsymbol{\rho}}' = \mathbf{q} \otimes \overline{\boldsymbol{\rho}} \otimes \mathbf{q}^{*} = Q^{+}Q^{-\mathrm{T}}\overline{\boldsymbol{\rho}} \tag{75}$$

This process is equivalent to rotating the vector $\boldsymbol{\rho}$ using a rotation matrix. The rotation matrix R_{q} is parameterized using the quaternion components and is defined as [6]:

$$R_{\mathrm{q}} = \mathrm{I}_{34} Q^{+} Q^{-\mathrm{T}} {I_{34}}^{\mathrm{T}} \tag{76}$$

where I_{34} is an identity matrix defined as:

$$I_{34} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{77}$$

This rotation matrix can be applied to the vector $\boldsymbol{\rho}$ such that the rotated vector $\boldsymbol{\rho}' = R_{\mathrm{q}}\boldsymbol{\rho}$.

With this quaternion and rotation matrix parameterization established, the angular velocity of the vehicle can be expressed as the time derivative of the quaternion state. It is

shown in [6] that the orientation of a system, given a gyroscope measurement can be calculated by:

$$\dot{\mathbf{q}} = \frac{1}{2} Q^{+}(\boldsymbol{\omega}_{\mathrm{m}} - \mathbf{b}_{\omega} + \boldsymbol{\eta}_{\omega \mathrm{m}}) \tag{78}$$

This leads to the overall state space model for the vehicle which is [51]:

$$\dot{\mathbf{x}} = \mathrm{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}) = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\mathbf{v}} \\ \dot{\mathbf{q}} \\ \dot{\mathbf{b}}_{\mathrm{a}} \\ \dot{\mathbf{b}}_{\omega} \end{bmatrix} = \begin{bmatrix} R_{\mathrm{q}} \mathbf{v} \\ \mathbf{f}_{\mathrm{m}} - \mathbf{b}_{\mathrm{a}} + R_{\mathrm{q}}^{\mathrm{T}} \mathbf{g}_{\mathbf{e}} + \boldsymbol{\eta}_{\mathrm{fm}} \\ 0.5 Q^{+}(\boldsymbol{\omega}_{\mathrm{m}} - \mathbf{b}_{\omega} + \boldsymbol{\eta}_{\omega \mathrm{m}}) \\ \boldsymbol{\eta}_{\mathrm{ba}} \\ \boldsymbol{\eta}_{\mathrm{b}\omega} \end{bmatrix} \tag{79}$$

Using this kinematic model, the physical states of the automobile system can be recursively predicted by integrating the IMU sensor data. Corrections for the filter are provided by secondary sensing devices which will be discussed in the following section.

4.2.3. **Sensor Measurement Models**

The INS filter uses a DGPS and orientation sensor for estimate correction. Both sensors are built into the OXTS RT3003 [54]. The measurement vector for the filter is given by:

$$\mathbf{y} = [\mathbf{y}_{\mathrm{p}}, \mathbf{y}_{\mathrm{q}}]^{\mathrm{T}} \tag{80}$$

where $\mathbf{y}_{\mathrm{p}}$ is the DGPS position measurement vector and $\mathbf{y}_{\mathrm{q}}$ is the vehicle orientation measurement vector represented using quaternions. The DGPS orientation is needed to improve the vehicle heading observability. Observability analysis for the system will be discussed in a later section. The GPS and orientation measurement models are:

$$\mathbf{y}_{\mathrm{p}} = \mathbf{p} + \boldsymbol{\eta}_{\mathrm{p}} \tag{81}$$

$$\mathbf{y}_{\mathrm{q}} = \mathbf{q} + \boldsymbol{\eta}_{\mathrm{q}} \tag{82}$$

where $\boldsymbol{\eta}_{\mathrm{p}}$ and $\boldsymbol{\eta}_{\mathrm{q}}$ are zero-mean Gaussian noise vectors such that $\boldsymbol{\eta}_{\mathrm{p}} \sim N(0, \boldsymbol{\sigma}_{\mathrm{p}}^2)$ and $\boldsymbol{\eta}_{\mathrm{q}} \sim N(0, \boldsymbol{\sigma}_{\mathrm{q}}^2)$; $\boldsymbol{\sigma}_{\mathrm{p}}^2$ and $\boldsymbol{\sigma}_{\mathrm{q}}^2$ are the variances of the GPS and orientation measurement noises respectively [51]. The measurement model noise vector is:

$$\boldsymbol{\nu} = \left[\boldsymbol{\eta}_{\mathrm{p}}, \boldsymbol{\eta}_{\mathrm{q}}\right]^{\mathrm{T}} \tag{83}$$

In the KITTI dataset, the roll, pitch and yaw of the vehicle body frame are provided [22]. The measurements have been converted to quaternion values in the estimator measurement model.

4.2.4. **Coordinate Frame Transformations**

The GPS sensor used in the KITTI data set operates using the WGS84 model [54]. To simplify the analysis, the GPS coordinates from the dataset were converted from the geodetic coordinates to a local tangent frame with a fixed origin. The geodetic coordinates are first converted into Earth-centered Earth-fixed (ECEF) coordinates before being transformed to the tangent plane. Figure 10 illustrates the three coordinate systems below.

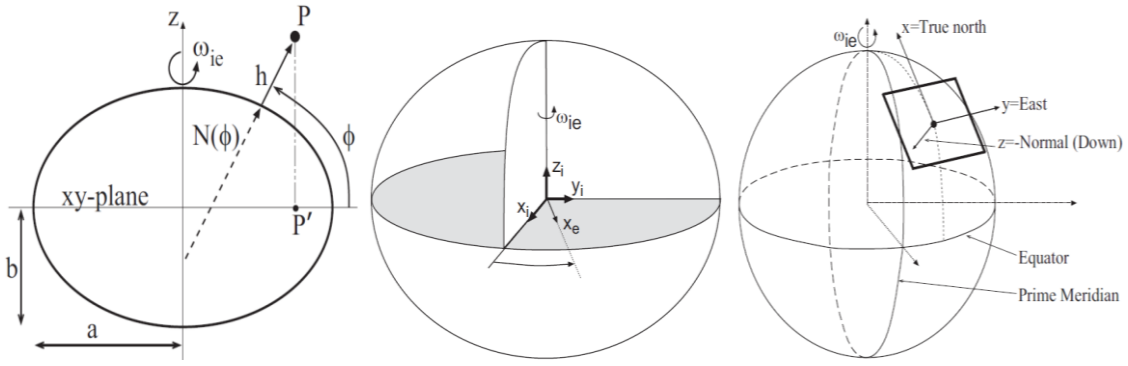


Figure 10 Geodetic, ECEF and Tangent Plane Coordinate Systems [6]

The geodetic coordinate system uses a reference ellipsoid to accurately approximate the geoid shape of the earth. In the WGS84 model, the reference ellipsoid can be defined by the parameters:

$$\mathrm{a} = 6378137\ \mathrm{m}$$

$$\frac{1}{\mathrm{f}} = 298.257223563$$

$$\mathrm{e} = \sqrt{\mathrm{f}(2-\mathrm{f})}$$

where a is the equatorial radius of the reference ellipse, f is the reference ellipse flatness and e is the eccentricity of the reference ellipse [6]. The meridian radius of a geodetic coordinate is defined as:

$$\mathrm{r_M}(\phi) = \frac{\mathrm{a}(1-\mathrm{e}^2)}{(1-\mathrm{e}^2\sin^2(\phi))^{\frac{3}{2}}} \tag{84}$$

where ϕ is the latitude of the point of interest [6]. Similarly, the prime normal radius of curvature is given by [6]:

$$r_N(\phi) = \frac{a}{(1 - e^2 \sin^2(\phi))^{\frac{1}{2}}} \tag{85}$$

Using the above relationships, the conversion from geodetic to ECEF coordinates is:

$$x = (r_N + h)\cos(\phi)\cos(\lambda) \tag{86}$$

$$y = (r_N + h)\cos(\phi)\sin(\lambda) \tag{87}$$

$$z = (r_N(1 - e^2) + h)\sin(\phi) \tag{88}$$

where h is the altitude and λ is the longitude of the point of interest [6]. The vector from a local tangent plane origin to an arbitrary point can be defined using:

$$\Delta\hat{\mathbf{x}}^e = \mathbf{P}^e - \mathbf{P}_0^e = [x, y, z]^e - [x_0, y_0, z_0]^e \tag{89}$$

$\mathbf{P}_0^e$ is the origin of the local tangent plane represented in the ECEF frame and $\mathbf{P}^e$ is a vector to an arbitrary point in the ECEF frame [6]. The difference between these coordinates produces the vector $\Delta\hat{\mathbf{x}}^e$ which is a vector that points from the tangent plane origin to the arbitrary point $\mathbf{P}^e$. The rotation matrix that rotates a vector from the ECEF frame to the local tangent plane is given by [6]:

$$R_{\mathbf{e}}^{t} = \begin{bmatrix} -\sin(\phi)\cos(\lambda) & -\sin(\phi)\sin(\lambda) & \cos(\phi) \\ -\sin(\lambda) & \cos(\lambda) & 0 \\ -\cos(\phi)\cos(\lambda) & -\cos(\phi)\sin(\lambda) & -\sin(\phi) \end{bmatrix} \tag{90}$$

Finally, the vector that defines a point with respect to the tangent frame, represented in the tangent plane is:

$$\mathbf{P}^{\mathrm{t}} = R_{\mathbf{e}}^{\mathrm{t}} \Delta \hat{\mathbf{x}}^{\mathrm{e}} \tag{91}$$

Using this process, the GPS data provided by the OXTS RT3003 was converted to local tangent plane coordinates. The first point in the dataset was used as the origin reference coordinate and was offset by the 0.93 m height of the GPS unit above ground level. Readers are directed to [6] for further information on this coordinate transformation.

4.2.5. **Observability Analysis**

Observability is the standard measure of a system's ability to determine state values, given the system inputs and available sensor data [38]. When a system is fully observable, all states can be solved given the current inputs and sensor information at that instant in time [38]. Observability status may change when states have values that render other states unobservable or sensor availability changes. Typically, a system is evaluated using many test scenarios to determine when or if states become unobservable during operations.

The approach for determining the observability of a linear system involves constructing the observability matrix for the system using the system matrix F and output matrix H [38]. The observability matrix $\mathcal{O}$ is:

$$\mathcal{O} = [H\ HF\ HF^2 \cdots HF^{n-1}]^{\mathrm{T}} \tag{92}$$

where n is the number of system states. A system is fully observable when this matrix is full rank [38]. The rank of this matrix was found to be sixteen for the designed INS,

however, since the system is nonlinear, this metric is not adequate for determining system observability.

Another approach for determining system observability that applies to nonlinear systems involves rewriting the state space model in its noise-free affine form and using Lie derivatives to construct the observability matrix [55]. The affine form of a system model has the structure:

$$\dot{\mathbf{x}} = f_0(\mathbf{x}) + f_1(\mathbf{x})\mathbf{u}_1 + f_2(\mathbf{x})\mathbf{u}_2 + \cdots + f_n(\mathbf{x})\mathbf{u}_n \tag{93}$$

where n is the total number of system inputs [55]. For the designed INS system, the noise-free affine form is:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\mathbf{v}} \\ \dot{\mathbf{q}} \\ \dot{\mathbf{b}}_{\mathrm{a}} \\ \dot{\mathbf{b}}_{\omega} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{g_e} - R_{\mathrm{q}}\mathbf{b}_{\mathrm{a}} \\ -0.5Q^{+}\mathbf{b}_{\omega} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ R_{\mathrm{q}} \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{f}_{\mathrm{m}} + \begin{bmatrix} 0 \\ 0 \\ 0.5Q^{+} \\ 0 \\ 0 \end{bmatrix} \boldsymbol{\omega}_{\mathrm{m}} \tag{94}$$

$$f_0 = \begin{bmatrix} \mathbf{v} \\ \mathbf{g_e} - R_{\mathrm{q}}\mathbf{b}_{\mathrm{a}} \\ -0.5Q^{+}\mathbf{b}_{\omega} \\ 0 \\ 0 \end{bmatrix}, \quad f_1 = \begin{bmatrix} 0 \\ R_{\mathrm{q}} \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad f_2 = \begin{bmatrix} 0 \\ 0 \\ 0.5Q^{+} \\ 0 \\ 0 \end{bmatrix} \tag{95}$$

For convenience when taking the Lie derivatives, the velocity has been expressed in the world frame.

The observability matrix is constructed by taking the Lie derivatives of the noise-free measurement function $h(\mathbf{x})$ with respect to the components of the affine system model $f_n(\mathbf{x})$ [55]. The following demonstrates the implementation of this process.

Considering the system when the measurement model only contains the DGPS readings:

$$\mathbf{y} = h(\mathbf{x}) = \mathbf{p} \tag{96}$$

The zeroth Lie derivative is simply [55]:

$$\mathcal{L}^0 h(\mathbf{x}) = h(\mathbf{x}) = \mathbf{p} \tag{97}$$

The gradient of this Lie derivative is:

$$\nabla \mathcal{L}^0 h(\mathbf{x}) = \frac{\partial \mathcal{L}^0 h(\mathbf{x})}{\partial \mathbf{x}} = [I_{3\times 3} \quad 0_{3\times 13}] \tag{98}$$

This matrix has a rank of 3 and spans $\mathbf{p}$ indicating that position is observable using this equation. The higher order Lie derivatives can be calculated recursively such that [55]:

$$\mathcal{L}_{f_j}^{i+1} h(\mathbf{x}) = \nabla \mathcal{L}^i h(\mathbf{x}) \cdot f_j, \{i \mid i \in \mathbb{R}, i \geq 0\} \tag{99}$$

Furthermore, higher order mixed Lie derivatives with respect to different functions in the affine form of the system model can be calculated using [55]:

$$\mathcal{L}_{f_j f_k}^{i+1} h(\mathbf{x}) = \nabla \mathcal{L}_{f_j}^{i} h(\mathbf{x}) \cdot f_k, \{i \mid i \in \mathbb{R}, i \geq 0\} \tag{100}$$

Using these properties, all possible Lie derivatives for this system can be obtained to generate a complete observability matrix [55]. This process can either be performed exhaustively using computer software or more efficiently by inspecting only the Lie derivatives that will yield enough linearly independent columns to make the observability matrix full rank. Using this constraint, the Lie derivatives that contain all elements equal

to zero are excluded. The following observability matrix is found to be full rank for the INS:

$$\mathcal{O} = \begin{bmatrix} \nabla\mathcal{L}^0 h(\mathbf{x}) \\ \nabla\mathcal{L}^1_{f_0} h(\mathbf{x}) \\ \nabla\mathcal{L}^2_{f_0 f_1} h(\mathbf{x}) \\ \nabla\mathcal{L}^2_{f_0 f_0} h(\mathbf{x}) \\ \nabla\mathcal{L}^3_{f_0 f_1 f_0} h(\mathbf{x}) \end{bmatrix} \tag{101}$$

The remainder of the observability matrix generation process proceeds as follows:

$$\boldsymbol{\mathcal{L}}^1_{f_0} h(\mathbf{x}) = \nabla\boldsymbol{\mathcal{L}}^0 h(\mathbf{x}) \cdot f_0 \tag{102}$$

The gradient of this Lie derivative is:

$$\nabla\mathcal{L}^1_{f_0} h(\mathbf{x}) = \frac{\partial \nabla\mathcal{L}^0 h(\mathbf{x}) \cdot f_0}{\partial \mathbf{x}} = [0_{3\times3} \quad I_{3\times3} \quad 0_{3\times10}] \tag{103}$$

This matrix is rank 3 and spans **v** indicating that the velocity can be determined using this equation. The next Lie derivative evaluated is:

$$\mathcal{L}^2_{f_0 f_1} h(\mathbf{x}) = \nabla\mathcal{L}^1_{f_0} h(\mathbf{x}) \cdot f_1 = R_{\mathrm{q}} \tag{104}$$

To evaluate the gradient of the rotation matrix R_{q}, it must first be converted into a 9×1 column vector using the elementary vectors $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ defined as [55]:

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{105}$$

The Lie derivative in equation (104) becomes [55]:

$$\mathcal{L}^2_{f_0 f_1} h(\mathbf{x}) = \begin{bmatrix} R_{\mathrm{q}} e_1 \\ R_{\mathrm{q}} e_2 \\ R_{\mathrm{q}} e_3 \end{bmatrix} = R_{\mathrm{q}} \mathbf{e}_i, \forall\, i = 1 \ldots 3 \tag{106}$$

Considering the quaternion form of the vector $\mathbf{e}_i$ such that $\bar{\mathbf{e}}_i = (0\ \mathbf{e}_i)^{\mathrm{T}}$, the right quaternion-product matrix $[\bar{\mathbf{e}}_i]^-$ can be generated for the vector $\bar{\mathbf{e}}_i$ using the same process presented in equation (73). Quaternion differentiation can be achieved using the following relation [6]:

$$\frac{\partial}{\partial \mathbf{q}}(\mathbf{q} \otimes \bar{\boldsymbol{\rho}} \otimes \mathbf{q}^*) = 2Q^{-\mathrm{T}}[\bar{\boldsymbol{\rho}}]^- \tag{107}$$

Recognizing that $\mathbf{q} \otimes \bar{\mathbf{e}}_i \otimes \mathbf{q}^*$ is equivalent to $R_{\mathrm{q}}\mathbf{e}_i$, the gradient of the Lie derivative in equation (106) is:

$$\nabla \mathcal{L}^2_{f_0 f_1} h(\mathbf{x}) = [0_{9\times 6} \quad 2I_{34}\mathrm{Q}^{-\mathrm{T}}[\bar{\mathbf{e}}_i]^- \quad 0_{9\times 6}] \tag{108}$$

Where $2I_{34}Q^{-\mathrm{T}}[\bar{\mathbf{e}}_i]^-$ is a 9×4 matrix with a rank of 4 that spans $\mathbf{q}$ indicating that the orientation of the system can be determined using this equation. The remaining vector space that is unobservable at this point spans $\mathbf{b}_\omega$ and $\mathbf{b}_\mathrm{a}$. Since the vector space for $\mathbf{p}$, $\mathbf{v}$ and $\mathbf{q}$ is spanned by the previously determined Lie derivatives in the observability matrix, the remaining Lie derivative gradients will be calculated with respect to $\mathbf{b}_\omega$ and $\mathbf{b}_\mathrm{a}$ only, to present the resulting observability matrix entries more compactly. The next Lie derivative required is:

$$\mathcal{L}^2_{f_0 f_0} h(\mathbf{x}) = \nabla \mathcal{L}^1_{f_0} h(\mathbf{x}) \cdot f_0 = \mathbf{g}_\mathbf{e} - R_{\mathrm{q}}\mathbf{b}_\mathrm{a} \tag{109}$$

The gradient of this Lie derivative with respect to $\mathbf{b}_\omega$ and $\mathbf{b}_\mathrm{a}$ is:

$$\nabla_{\mathbf{b}_\omega, \mathbf{b}_\mathrm{a}} \mathcal{L}^2_{f_0 f_0} h(\mathbf{x}) = [0_{3\times 3} \quad -R_{\mathrm{q}}] \tag{110}$$

This matrix is rank 3 and spans $\mathbf{b}_a$ indicating that the accelerometer bias is observable using this equation. The final Lie derivate needed is:

$$\mathcal{L}^3_{f_0 f_1 f_0} h(\mathbf{x}) = \nabla \mathcal{L}^1_{f_0 f_1} h(\mathbf{x}) \cdot f_0 = -I_{34} Q^{-\mathrm{T}} [\bar{\mathbf{e}}_i]^- Q^+ \mathbf{b}_\omega \tag{111}$$

The gradient of this Lie derivative with respect to $\mathbf{b}_\omega$ and $\mathbf{b}_a$ is:

$$\nabla_{\mathbf{b}_\omega, \mathbf{b}_a} \mathcal{L}^3_{f_0 f_1 f_0} h(\mathbf{x}) = [-I_{34} Q^{-\mathrm{T}} [\bar{\mathbf{e}}_i]^- Q^+ \quad 0_{9\times 3}] \tag{112}$$

This matrix is rank 3 and spans $\mathbf{b}_\omega$ indicating that the gyroscope bias is observable using this equation. These calculations prove the claim that the observability matrix in equation (101) is full rank and thus the system states should be locally weakly observable (according to Theorem 3.1 in [56]) given a measurement model that provides position [55]. Local weak observability relates to the ability of a system state to be distinguishable when initialized in a close neighbourhood of its true value without needing a considerable amount of time to stabilize to the true state [56]. It was found experimentally that without heading measurements, the filter diverges quickly due to instabilities. This was not further investigated in this thesis since the experiments performed had access to compass heading measurements to overcome this issue. However, it is important to note that following a suitable initialization and careful tuning procedure should allow the filter to operate without heading measurements, as indicated by the observability study. For the experiments of this thesis, a heading measurement was always used for more reliable operation of the filters.

4.3. Vehicle Inertial Navigation Experiment

4.3.1. Inertial Navigation System Filter Implementation

The vehicle trajectory used in this study was obtained in a residential area in Karlsruhe, Germany [22]. The vehicle drove in a looping path that involved two U-turn type manoeuvres which provided approximately 116 seconds of data measurements. The map of the area is shown below in Figure 11 with the blue line roughly outlining the path that the vehicle travelled. During INS testing and validation, random Gaussian noise was added to the DGPS data to mimic the reduced accuracy of GPS. Gaussian noise was also added to the heading measurements to test orientation tracking.

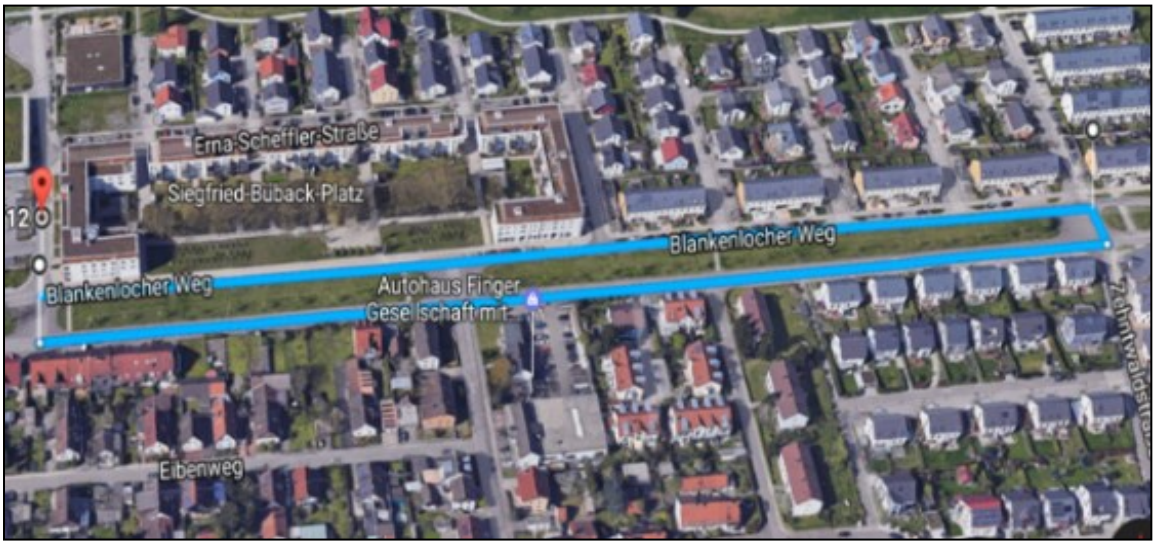


Figure 11 Map of Vehicle Trajectory in Karlsruhe, Germany [57]

The INS algorithm was developed and tested using MATLAB. The EKF linearized matrices were all pre-calculated using MATLAB's symbolic toolbox to reduce computation time while running the state estimator. For the OXTS RT3003 module, GPS

accuracy using Standard Positioning Service (SPS) or a Satellite-Based Augmentation System (SBAS) is $1.5\ m$ and $0.6\ m$ respectively [54]. For the simulations, the uncertainty in the GPS measurements was modelled as random additive Gaussian noise ranging from $0\ m$ to $1\ m$ in both the x and y directions, and ranging from $0\ m$ to $0.2\ m$ in the z direction. This gives an approximate Euclidean position accuracy range of $0\ m$ to $1.48\ m$. The z direction was assigned less uncertainty using the assumption that the vehicle altitude will not change substantially in a $0\ m$ to $1\ m\ x, y$ area on a smooth road profile. Similarly, noise is added to the roll, pitch, yaw measurements provided in the data set to introduce some uncertainty to these values. Random Gaussian noise is added to each measurement ranging from approximately 0 to 3^0. The original measurements without added noise are used as the true orientation angles during analysis.

The IMM algorithm was implemented as shown in Figure 12. The two EKF models used within the IMM are designed to contain differently tuned measurement and process noise parameters. The first EKF filter was tuned by testing sets of process and measurement noise parameters that minimized the root mean square (RMS) Euclidean position error and RMS orientation angle errors. These parameters were used for the single EKF filter and for the first mode of the IMM filter during testing. Keeping the noise parameters of the first mode fixed, this process was repeated to select noise parameters for the second mode of the IMM filter. The switching matrix probabilities for transitioning from one mode to another were both set to 3%. Both initial model probabilities were set to 50% since the system behaviour is unpredictable until the filter stabilizes. In certain instances, the likelihood of either model may approach zero causing matrix singularities. To address this

problem in the MATLAB simulator, conditions have been set such that if the likelihood of one model approaches zero while the other has a finite value, then the likelihood of the finite value model is assigned a magnitude of 1 while the other is given 0 weight.

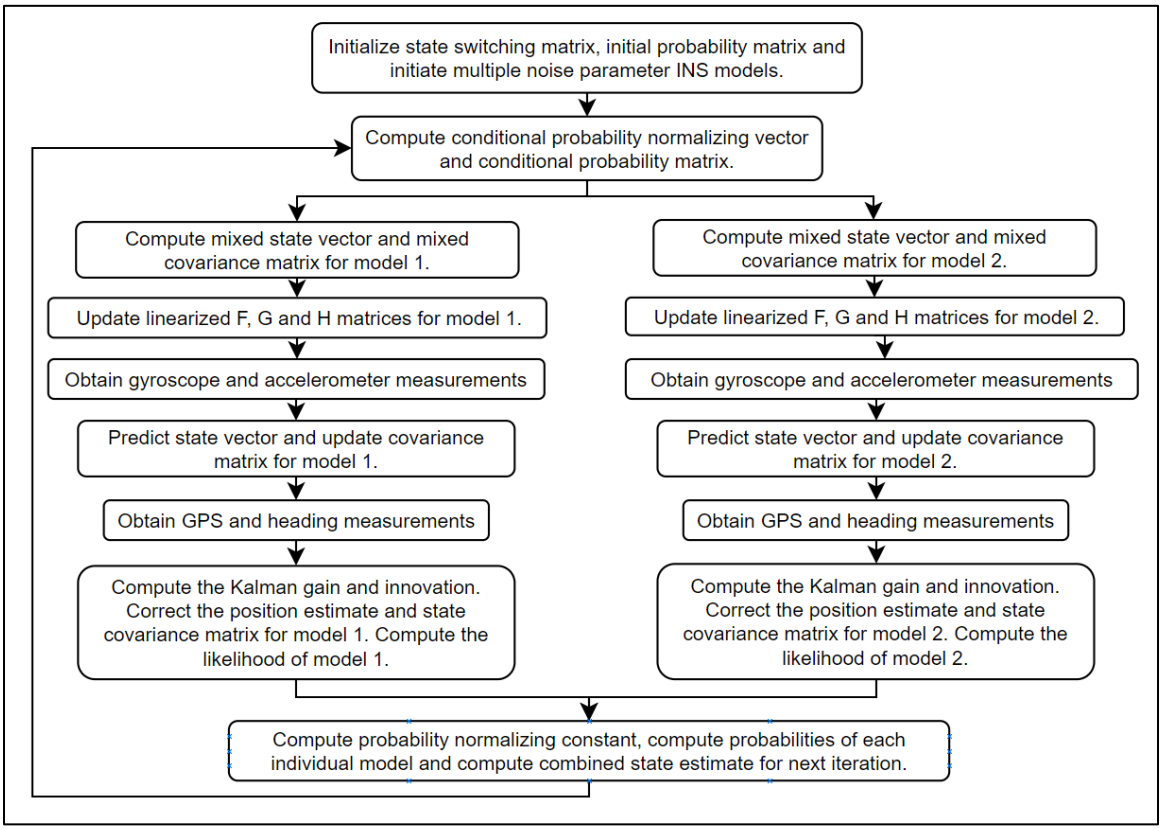


Figure 12 Interactive Multiple Model Filter Vehicle INS Implementation

4.3.2. **Experimental Results and Analysis**

The INS system was tested using both the single EKF and IMM filter for comparison. For convenience, since differences in the results of both algorithms are not visibly apparent, only the IMM filter plots are presented except for Figure 13 which shows both estimated trajectories.

The position estimate results shown in Figure 13 validate the accuracy of the IMM two mode state predictor. The DGPS ground truth model is represented by the blue line while the IMM and EKF prediction results are indicated by the red and green lines respectively. Both the EKF and IMM algorithms were able to accurately predict the position of the vehicle during all sequences of the trajectory. The IMM estimates tend to transition more abruptly than the EKF predictions because of the switching between modes. It is evident that the IMM tracker tends to predict the path of the vehicle better than the EKF filter during the two U-turn manoeuvres shown near positions $(-425\ m, -42\ m)$ and $(228\ m, 13\ m)$. In both cases, the EKF algorithm predicts that the vehicle followed a trajectory on the inside of the true vehicle path while the IMM filter follows the true path more closely.

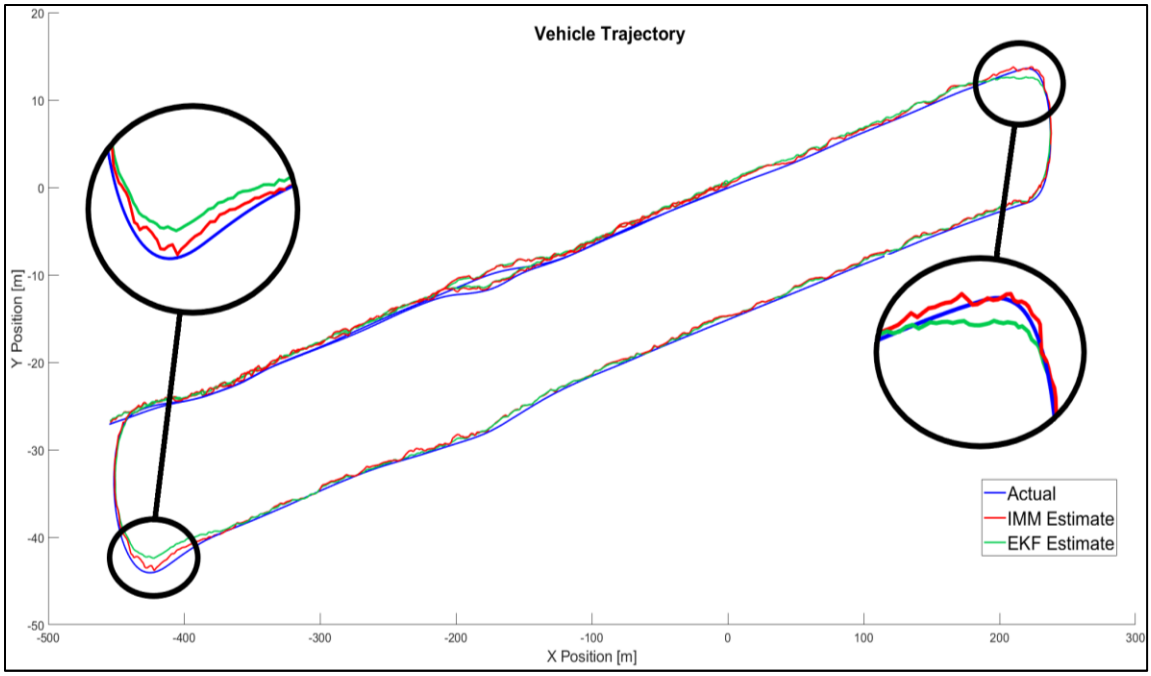


Figure 13 EKF and IMM Vehicle Trajectory Estimates

The IMM filter generally tracks the position of the vehicle well in all three directions as shown in Figure 14. To evaluate the EKF and IMM filters, the RMS Euclidean positional error during the full tracking sequence was calculated for twenty simulations. The EKF filter produced an average positional error of $0.8001\ m$ while the IMM filter gave an average error of $0.7553\ m$. This is an average positional prediction improvement of $0.0448\ m$ for this trajectory. Based on this, applying the IMM filter to track a system undergoing more turning manoeuvres would potentially reduce a large RMS position error that is likely to occur if a single EKF is used. Furthermore, the IMM filter can still be improved and refined by using more than two models or tuning the noise parameters.

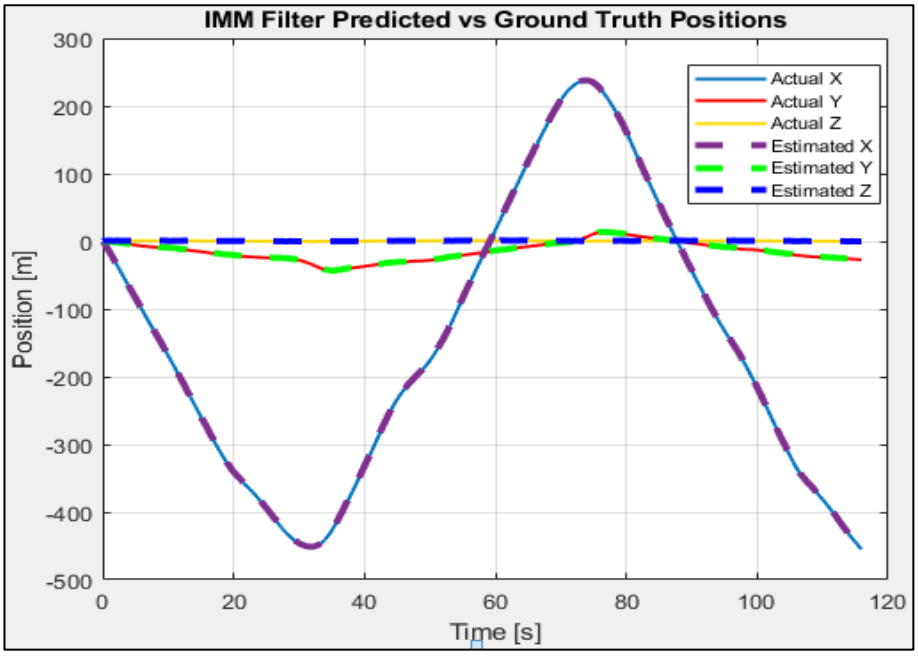


Figure 14 Vehicle IMM Filter Position Estimates

Figure 15 shows the IMM estimated position errors during the simulation. The 95% confidence bounds of each error vector is calculated by multiplying the combined

covariance matrix elements (from Eq. (36)) for each position state by a factor of two. In general, all position errors are below the confidence bound. The position error in the z direction is below the error bound for all simulated time, while the x and y errors periodically exceed the bounds during times when the vehicle is turning or highly erroneous GPS measurements are provided.

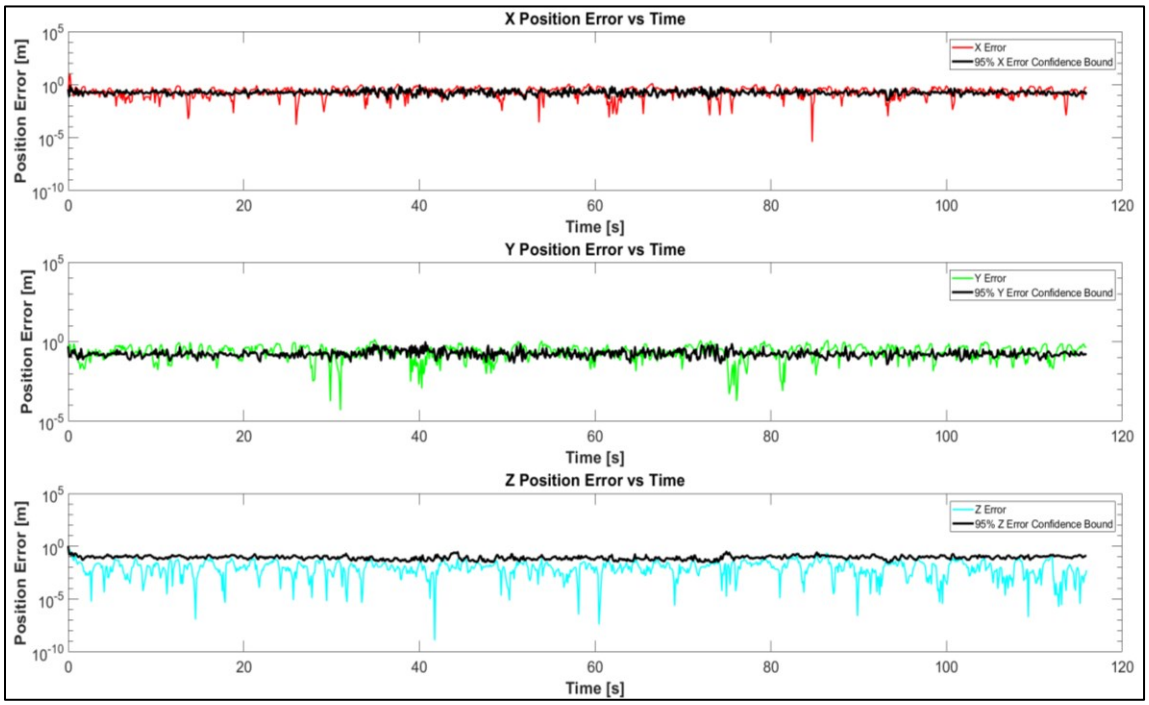


Figure 15 Vehicle IMM Filter Position Error

The velocity estimates from the IMM filter give expected results that are shown below in Figure 16. The x direction of the vehicle points along its forward axis, therefore, the vehicle velocity is generally forward during all time instances with varying speeds. The y and z axes point leftward and upward on the vehicle respectively, therefore, these velocity components are generally small. The Ackerman design of the vehicle steering system causes the instantaneous center of rotation for the vehicle to be located somewhere

in the direction of the inside of the turn. As the vehicle performs the turning motion, the velocity vector is generally not perfectly tangential to the turning path and there is likely tire deformation from the lateral friction load that causes slippage [13]. As a result, there are instances during the turn where the velocity vector has components in both the x and y directions as shown at times $34\ s$ and $76\ s$. If steering data were included with the data set, the Ackermann motion model could have been included in the IMM design to improve these estimates.

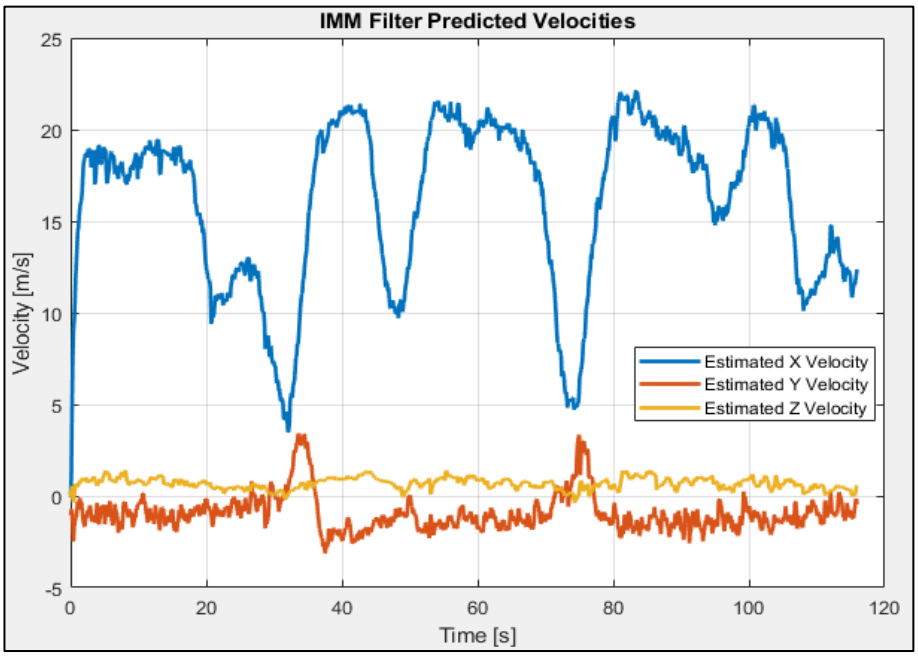


Figure 16 Vehicle IMM Filter Velocity Estimates

The orientation angle predictions obtained using the IMM filter are shown in Figure 17. The IMM filter accurately tracks all roll, pitch and yaw angles during the simulation. Roll and pitch generally remained constant throughout the duration of the vehicle trajectory while the yaw showed large variations during turning manoeuvres. The mean orientation

angle error for both EKF and IMM were approximately 3.055^0 with negligible difference between either result. With further noise parameter turning or the inclusion of more than two filter models, it is likely that the IMM can yield better results.

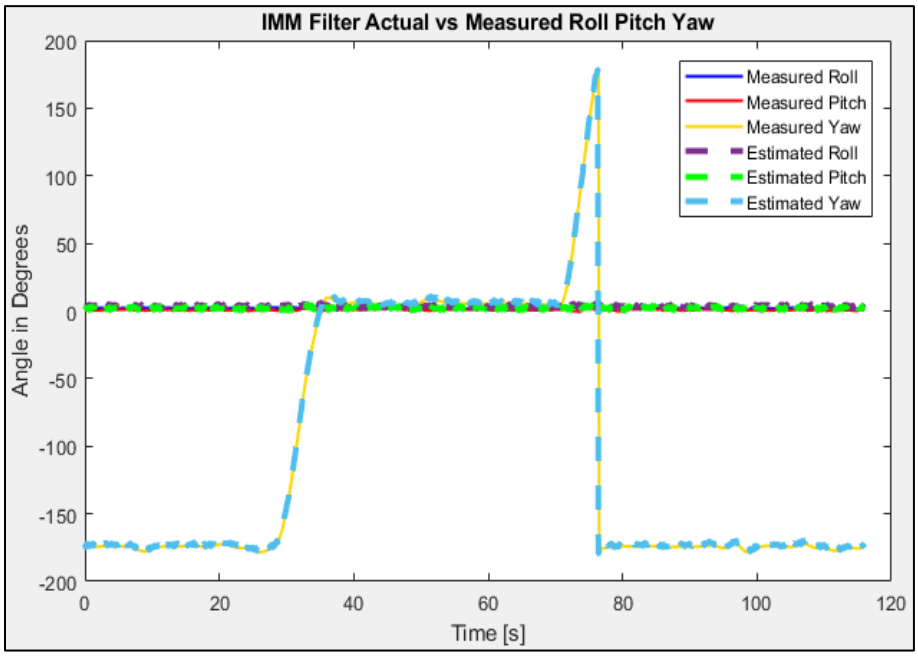


Figure 17 Vehicle IMM Filter Roll, Pitch and Yaw Estimates

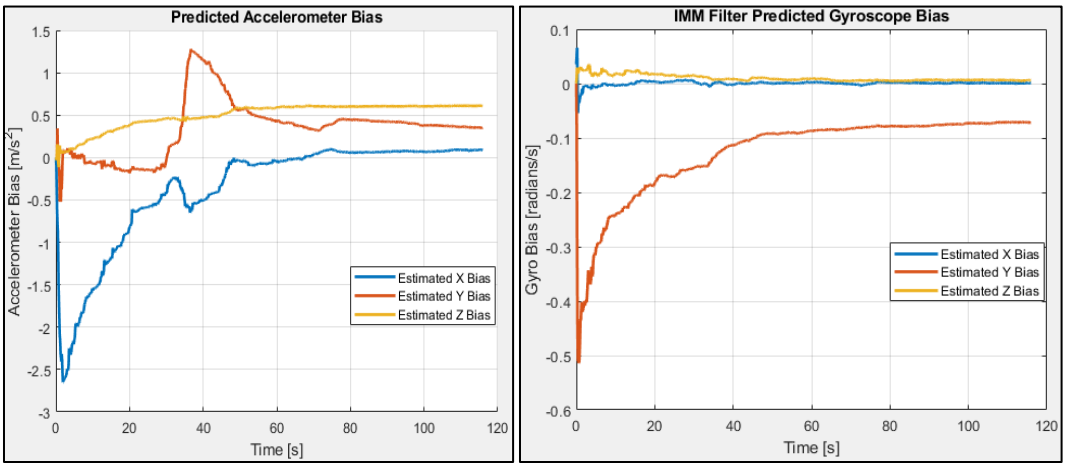


Figure 18 Vehicle IMM Filter Accelerometer and Gyroscope Bias Estimates

The IMM filter accelerometer and gyroscope biases are shown below in Figure 18. The accelerometer bias in the z direction does not experience drastic fluctuations during simulation, while the x and y biases tend to have steep transitions during the turning manoeuvres. The accelerometer biases have large spikes initially that tend to settle the longer the simulation runs. The gyroscope biases all show rapid transitions as the system starts up and then settle to relatively steady state values after the vehicle has been moving for approximately 40 seconds.

Figure 19 shows the probabilities of each EKF filter mode in the IMM for the duration of the simulation. The model used in the single EKF filter typically has a higher probability than model 2, however, model 2 does improve the INS performance with its state estimate contribution.

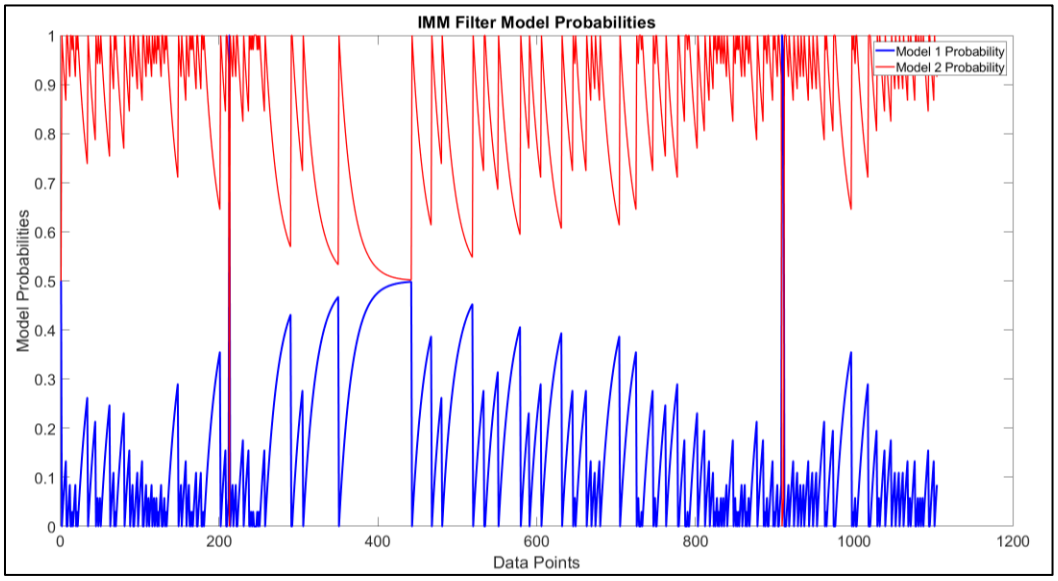


Figure 19 Vehicle IMM Filter Model Probabilities

4.4. Conclusions

This chapter has presented an INS system for predicting the states of an automobile driving around a looping path. The EKF algorithm is shown to be a valid method for predicting the dynamics of a nonlinear state-space model through the process of linearization. Incorporating multiple EKF models within an IMM algorithm has been shown to improve the tracking accuracy of the INS system. This is most visibly apparent when the vehicle performs a turn during the trajectory. The IMM mode switching and prediction mixing allows the estimator to adaptively adjust the noise parameters during straightaways or turning manoeuvres.

The extent of the tracking performance improvements through noise tuning can be further explored by additional parameter adjustments that may reduce localization error. Adding more models to the IMM filter bank that contain additional noise figures can potentially improve filter performance at the expense of higher computational demand. To test this hypothesis, a dataset from an automobile that performs many consecutive turning manoeuvres would be required. This would provide better insight into IMM filter performance versus single model filters for highly dynamic trajectories. Furthermore, the inclusion of multiple system models, as demonstrated in section 3.2.2 could potentially improve IMM localization. The vehicle under study in the KITTI Vision Benchmark dataset has an Ackermann steering configuration, therefore, including this steering system geometry in another IMM mode may improve tracking. This type of model typically

requires the inclusion of vehicle steering angle as feedback; therefore, it was not included in this experiment due to that data being unavailable.

Chapter 5

Skid-Steer Robot Inertial Navigation System

About this chapter: This chapter analyses an INS estimation system for a skid-steer mobile robot. The chapter examines the performance of the filter developed in Chapter 4 when applied to the Seekur Jr mobile robot. The IMM has been redesigned to include a skid-steer motion model that tracks the instantaneous centers of rotations of the robot to predict slippage and improve state tracking. The chapter also presents the design of an experimental testbed for the purpose of multi-model estimation research and validates an IMM filter design for navigation of the Seekur Jr mobile robot.

5.1. Problem Formulation

Several steering configurations exist for mobile robots that allow for many different motion trajectories [7]. One popular method of steering that is used extensively in mobile robot designs is the skid steer configuration [58]. This configuration relies on lateral wheel slippage that allows the robot to make turning manoeuvres. The wheel-ground interactions associated with this type of steering are complex and cannot be modelled easily [59]. This leads to difficulty when developing a state estimator that can accurately track the system when unmodeled slippage occurs [58]. Many methods for dealing with this problem exist,

however, often they require many physical parameters to be known which may not be readily available or may change as the system operates [59]. For example, dynamic models can predict the longitudinal and lateral forces experienced by robot wheels during a skid [60]. However, if the ground terrain changes then the friction between the ground and wheel will also change leading to estimation errors [60].

The approach that has been selected to address this problem involves adding a second motion model to the IMM filter that tracks the ICRs of the left and right side of the robot. This model allows the lateral velocity of the robot to be tracked more accurately which provides detection of slipping that cannot be as apparently captured with the vehicle model in 4.2.2. The experiment has been performed using the Seekur Jr robot in the ISLAB at Memorial University of Newfoundland.

5.2. **Methodology**

5.2.1. **Skid Steer Robot Kinematics**

Skid-steer is a steering configuration that requires lateral wheel slippage to perform a turning manoeuvre [60]. Instead of an actuated steering mechanism that changes the direction of the robot wheels, skid-steer turns the robot by rotating the wheels on each side at different speeds and/or directions to induce a turn [59]. This allows the robot to rotate in place for efficient manoeuvres in confined spaces. One main drawback of this configuration is that the slippage of the wheels is difficult to model, especially on dynamic

outdoor terrains [60]. This makes it challenging to predict the motion of these systems accurately.

Many methods exist that model the complex ground to wheel interactions arising in skid-steer configurations, however, these models typically require dynamic physical parameters of the robot such as tire deflection and terrain friction coefficient [60]. These are continuously varying parameters that cause the lateral forces on the robot wheels to fluctuate, making state prediction complicated [60].

Another approach to modelling the skid-steer configuration considers the locations of the ICRs for the left and right side of the robot as changing vectors [58]. These vectors are added to the vehicle state space model to account for the lateral slippage of the skid-steer robot during turn manoeuvres. Considering the case of a 2D skid-steer mobile robot as depicted in Figure 20. The instantaneous center of rotation of the robot body relative to the ground is given by the coordinates $(\mathbf{x}_{\mathrm{ICR}}, \mathbf{y}_{\mathrm{ICR}})$. The Y coordinate of this point can be calculated by:

$$\mathbf{y}_{\mathrm{ICR}} = \frac{\mathbf{v}_{\mathrm{x}}}{\boldsymbol{\omega}_{\mathrm{z}}} \tag{113}$$

where $\mathbf{v}_{\mathrm{x}}$ is the velocity of the robot in the X direction and $\boldsymbol{\omega}_{\mathrm{z}}$ is the angular velocity of the robot about the Z-axis (the Z-axis points out of the page in Figure 20) [58].

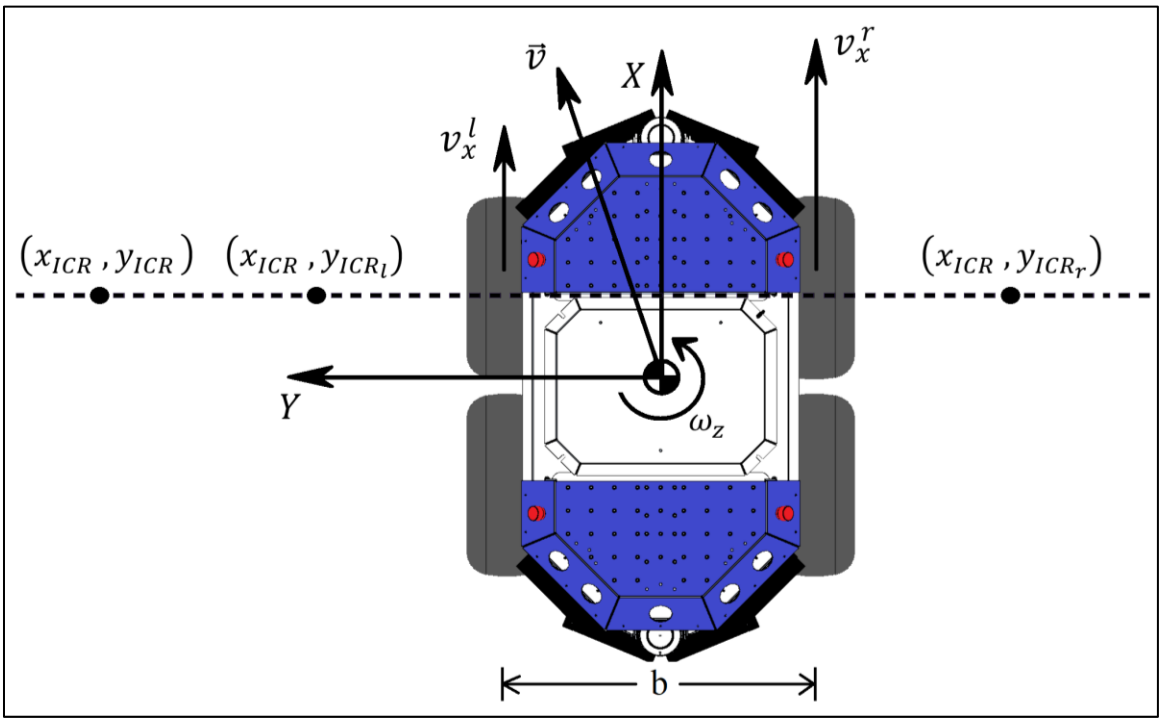


Figure 20 Skid-steer Kinematics 2D Robot [61]

The individual wheel velocities of the left and right side of the robot with respect to the robot body are denoted as $\mathbf{v}_x^l$ and $\mathbf{v}_x^r$ respectively. Using these wheel velocities, the Y coordinates of the ICRs for the left and right side of the robot are:

$$\mathbf{y}_{\mathrm{ICR_l}} = \frac{\mathbf{v}_x^l - \mathbf{v}_x}{\boldsymbol{\omega}_z} \tag{114}$$

$$\mathbf{y}_{\mathrm{ICR_r}} = \frac{\mathbf{v}_x^r - \mathbf{v}_x}{\boldsymbol{\omega}_z} \tag{115}$$

respectively [58]. The X coordinate of the robot body ICR with respect to the ground is:

$$\mathbf{x}_{\mathrm{ICR}} = \frac{-\mathbf{v}_y}{\boldsymbol{\omega}_z} \tag{116}$$

where $\mathbf{v}_y$ is the robot lateral velocity in the Y direction [58]. It is shown [59] that all the described ICR coordinates lie on a line parallel to the robot Y axis, therefore:

$$\mathbf{x}_{\mathrm{ICR_r}} = \mathbf{x}_{\mathrm{ICR_l}} = \mathbf{x}_{\mathrm{ICR}} = \frac{-\mathbf{v}_y}{\boldsymbol{\omega}_z} \tag{117}$$

Manipulating equations (114), (115) and (116) to isolate the body linear and angular velocities in terms of the ICR coordinates and wheel velocities gives:

$$\mathbf{v}_x = \frac{\mathbf{v}_x^l \mathbf{y}_{\mathrm{ICR_r}} - \mathbf{v}_x^r \mathbf{y}_{\mathrm{ICR_l}}}{-\left|\mathbf{y}_{\mathrm{ICR_r}} - \mathbf{y}_{\mathrm{ICR_l}}\right|} \tag{118}$$

$$\mathbf{v}_y = \frac{(\mathbf{v}_x^r - \mathbf{v}_x^l)\mathbf{x}_{\mathrm{ICR}}}{-\left|\mathbf{y}_{\mathrm{ICR_r}} - \mathbf{y}_{\mathrm{ICR_l}}\right|} \tag{119}$$

$$\boldsymbol{\omega}_z = -\frac{\mathbf{v}_x^r - \mathbf{v}_x^l}{-\left|\mathbf{y}_{\mathrm{ICR_r}} - \mathbf{y}_{\mathrm{ICR_l}}\right|} \tag{120}$$

The absolute values in the denominators of these equations are used to ensure filter convergence by avoiding division by zero and keeping the denominators negative [58].

Considering the case where $\mathbf{v}_y$ is constrained to be zero (no slip constraint), the coordinate $\mathbf{x}_{\mathrm{ICR}}$ also becomes zero and therefore lies along the Y-axis. This constraint is commonly imposed when modelling two-wheel differential-drive robots for odometry state estimators [7]. The differential-drive robot configuration is shown in Figure 21. Its kinematics are summarized below:

$$\mathbf{v}_x = \frac{\mathbf{v}_x^l + \mathbf{v}_x^r}{2} \tag{121}$$

$$\mathbf{v}_y = 0 \tag{122}$$

$$\boldsymbol{\omega}_z = \frac{\mathbf{v}_x^r - \mathbf{v}_x^l}{b} \tag{123}$$

where b is the distance between the robot wheels [58].

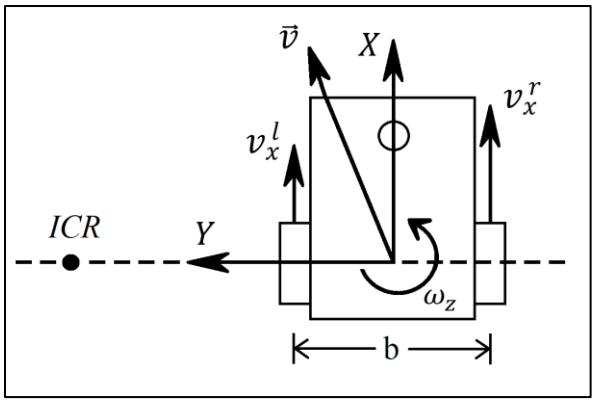


Figure 21 Differential-Drive Robot Kinematics

Experiments in [58] show that the estimation of ICR coordinates can improve skid-steer robot navigation over using differential-drive robot kinematics. The experiments also show that the ICR odometry is much better at maintaining robot localization during periods of GPS dropout over the differential-drive for wheeled mobile robots [58]. Considering scenarios like the robot operating in GPS denied areas, the robot transitioning from an outdoor to an indoor environment, or other sensor systems becoming unreliable, the inclusion of the ICR kinematics in the IMM-INS designed in section 4.2.2 should improve state estimation performance when any of these conditions occur.

5.2.2. **Skid-Steer Robot Inertial Navigation System**

The motion of wheeled mobile robots is not always constrained to a 2-D horizontal plane. However, on relatively flat terrains this assumption is an adequate approximation

[7]. That considered, the position of the ICR locations along the Z-axis should not change substantially during operations. Using this assumption, the INS design for the vehicle model in section 4.2.2 can be modified to include skid-steer kinematics.

It has been shown in [58] that variations in the ICR coordinates can be modelled as random walk processes such that:

$$\dot{\mathbf{x}}_{\mathrm{ICR}} = \boldsymbol{\eta}_{\mathrm{x_{ICR}}}, \qquad \dot{\mathbf{y}}_{\mathrm{ICR_l}} = \boldsymbol{\eta}_{\mathrm{y_{ICR_l}}}, \qquad \dot{\mathbf{y}}_{\mathrm{ICR_r}} = \boldsymbol{\eta}_{\mathrm{y_{ICR_r}}} \tag{124}$$

where $\boldsymbol{\eta}_{\mathrm{x_{ICR}}}$, $\boldsymbol{\eta}_{\mathrm{y_{ICR_l}}}$, and $\boldsymbol{\eta}_{\mathrm{y_{ICR_r}}}$, are zero-mean Gaussian noise vectors with variances $\boldsymbol{\sigma}^2_{\mathrm{x_{ICR}}}$, $\boldsymbol{\sigma}^2_{\mathrm{y_{ICR_l}}}$, and $\boldsymbol{\sigma}^2_{\mathrm{y_{ICR_r}}}$, respectively.

The estimated states of the system are included in the following state vector:

$$\mathbf{x} = \left[\mathbf{p}, \mathbf{v}, \mathbf{q}, \mathbf{b}_{\mathrm{a}}, \mathbf{b}_{\omega}, \mathbf{x}_{\mathrm{ICR_l}}, \mathbf{y}_{\mathrm{ICR_l}}, \mathbf{y}_{\mathrm{ICR_r}}\right]^{\mathrm{T}} \tag{125}$$

Therefore, the skid-steer mobile robot state space model is:

$$\dot{\mathbf{x}} = \mathrm{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}) = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\mathbf{v}} \\ \dot{\mathbf{q}} \\ \dot{\mathbf{b}}_{\mathrm{a}} \\ \dot{\mathbf{b}}_{\omega} \\ \dot{\mathbf{x}}_{\mathrm{ICR}} \\ \dot{\mathbf{y}}_{\mathrm{ICR_l}} \\ \dot{\mathbf{y}}_{\mathrm{ICR_r}} \end{bmatrix} = \begin{bmatrix} R_{\mathrm{q}}\mathbf{v} \\ \mathbf{f}_{\mathrm{m}} - \mathbf{b}_{\mathrm{a}} + R_{\mathrm{q}}^{\mathrm{T}}\mathbf{g}_{\mathrm{e}} + \boldsymbol{\eta}_{\mathrm{fm}} \\ 0.5Q^{+}(\boldsymbol{\omega}_{\mathrm{m}} - \mathbf{b}_{\omega} + \boldsymbol{\eta}_{\omega\mathrm{m}}) \\ \boldsymbol{\eta}_{\mathrm{ba}} \\ \boldsymbol{\eta}_{\mathrm{b}\omega} \\ \boldsymbol{\eta}_{\mathrm{x_{ICR}}} \\ \boldsymbol{\eta}_{\mathrm{y_{ICR_l}}} \\ \boldsymbol{\eta}_{\mathrm{y_{ICR_r}}} \end{bmatrix} \tag{126}$$

The measurement model for this system includes DGPS positional feedback, magnetometer readings and wheel encoder readings from the left and right side of the robot that determine $\mathbf{v}_{\mathrm{x}}^{\mathrm{l}}$, $\mathbf{v}_{\mathrm{x}}^{\mathrm{r}}$ and $\boldsymbol{\omega}_{\mathrm{z}}$. The measurement vector for the system is given by:

$$\mathbf{y} = [\mathbf{y}_\mathrm{p}, \mathbf{y}_\mathrm{m}, \mathbf{v}_\mathrm{x}^\mathrm{l}, \mathbf{v}_\mathrm{x}^\mathrm{r}, \boldsymbol{\omega}_\mathrm{z}]^\mathrm{T} \tag{127}$$

where $\mathbf{y}_\mathrm{m}$ is the magnetometer measurement vector. The measurement model for the magnetometer is given as:

$$\mathbf{y}_\mathrm{m} = R_\mathrm{q}^\mathrm{T}\mathbf{m}_\mathrm{e} + \boldsymbol{\eta}_\mathrm{m} \tag{128}$$

where $\mathbf{m}_\mathrm{e}$ is the local magnetic field vector in the world frame and $\boldsymbol{\eta}_\mathrm{m}$ is zero-mean Gaussian noise corrupting the measurement such that $\boldsymbol{\eta}_\mathrm{m} \sim N(0, \boldsymbol{\sigma}_\mathrm{m}^2)$ with variance $\boldsymbol{\sigma}_\mathrm{m}^2$ [51]. The robot velocity measurements are obtained from Eq. (118)-(120).

5.2.3. **Differential Global Positioning Systems**

5.2.3.1. **Differential Global Positioning System Background**

Standard GPS systems encounter many sources of error when determining position. The main sources include ionospheric delay, tropospheric delay, ephemeris, clock errors and multipath signal reflection [62]. Ionospheric and tropospheric delays involve the slowing of satellite signal propagation due to complex signal interactions with the physical compositions of these atmospheric layers [6]. Ephemeris errors occur when a satellite has an orbital trajectory bias [6]. The satellite transmits incorrect ephemeris data since its orbit does not match its expected trajectory. Clock errors are the result of drift in the atomic clocks of satellites which cause an offset with respect to GPS receiver clocks [63]. Multipath error occurs when the transmitted satellite signal reflects off objects near the GPS receiver causing several extra delayed signals to be perceived by the receiver [63].

The DGPS configuration is an effective way to reduce the impact of multiple error sources in a GPS system. To implement this system, two GPS units operate in unison. One unit is designated as the stationary base station, while the second unit is the moving rover [64]. The exact position of the base station must be known accurately [64]. The base station receives Global Navigation Satellite System (GNSS) signals and calculates the pseudoranges to the visible satellites [65]. The pseudorange errors are calculated using the accurately known position of the base station [65]. The error corrections are transmitted to the rover unit via a radio signal [64]. The rover applies the pseudorange error corrections to the incoming GNSS signals to improve its positional estimate substantially [64]. This process is depicted below in Figure 22.

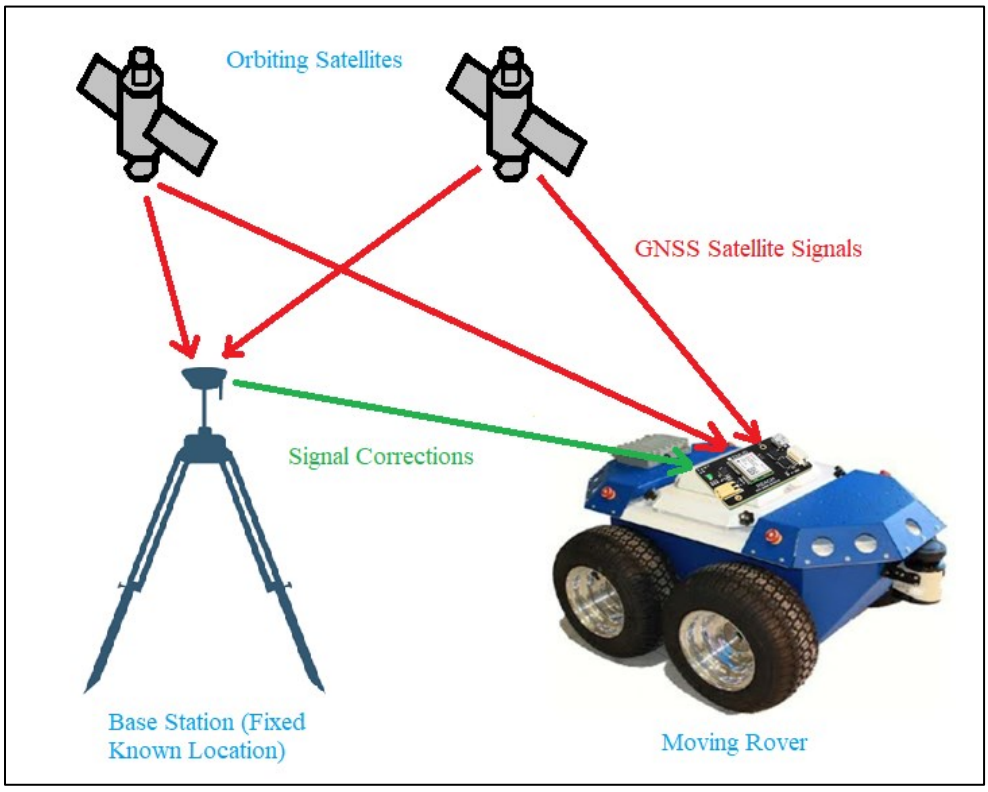


Figure 22 Differential Global Positioning System Overview [61] [24]

5.2.3.2. **Emlid Reach Differential Global Positioning System**

The Emlid Reach GPS module is an Intel Edison computer chip with an integrated IMU and GPS sensor. The Reach units used in the Seekur Jr experiments are the RTK GNSS modules shown in Figure 23.

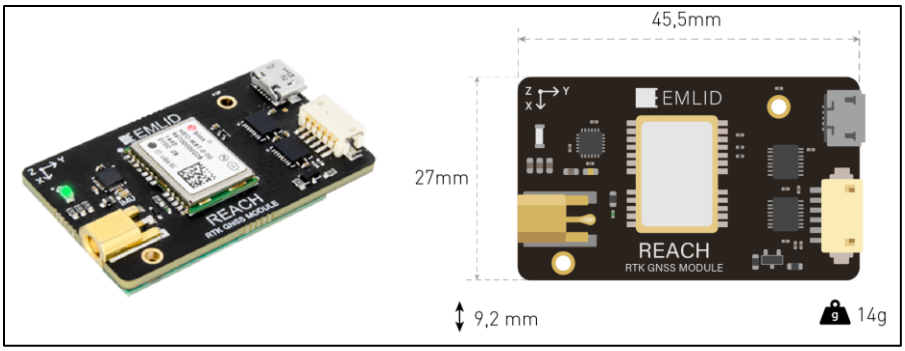


Figure 23 Emlid Reach RTK GNSS Module [24] [66]

For the DGPS application, two Reach units are used. One unit is fixed to a stationary tripod while the other is secured to the Seekur Jr robot. The position of the base station unit is determined by averaging its standalone GPS readings over a long period of time. This averaged positional value is then set as its fixed location in its configuration files. The DGPS correction process is completed directly on the Reach unit. The manufacturer uses the open-source RTK processing software known as RTKLIB for this procedure [64] [65].

The implemented base station and rover parameters are summarized in Table 5 and Table 6 respectively. The Reach units are programmed using the ReachView app shown in Figure 24. Parameters have been selected based on the recommendations from Emlid

support staff. Further details on the specific implementation and calibration of the Reach units will be discussed in section 5.2.4.4.

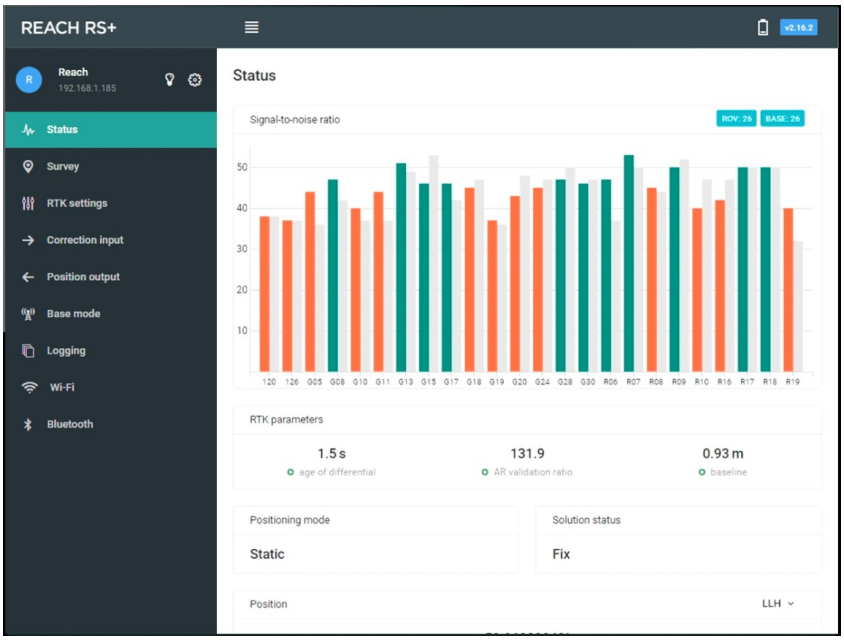


Figure 24 Emlid ReachView App [67]

Table 5 Reach Base Station Configuration

Reach Base Station Module		
Base Mode		
Corrections Output (Serial)	Device	Baud Rate
	UART	57600
Base Coordinates	Coordinates Input Mode	Coordinate Accumulation Time
	Average Single	5 Minutes
RTK Settings		
RTK	Positioning Mode	GPS AR Mode
	Static	Fix-and-hold
GLONASS AR Mode	Elevation Mask Angle	SNR Mask
On	15 Degrees	35 Degrees
Max Acceleration	Vertical	Horizontal
	1 m/s²	1 m/s²

Table 6 Reach Rover Configuration

Reach Rover Module		
RTK Settings		
RTK	Positioning Mode	GPS AR Mode
	Kinematic	Fix-and-hold
GLONASS AR Mode	Elevation Mask Angle	SNR Mask
On	15 Degrees	35 Degrees
Max Acceleration	Vertical	Horizontal
	1 m/s²	1 m/s²
Correction Input		
Base Correction (Serial)	Device	Baud Rate
	UART	57600
	Format	
	RTCM3	
Position Output		
Output 1 (TCP)	Role	Address
	Server	localhost
	Port	Format
	8889	LLH

5.2.4. **Seekur Jr Robot**

5.2.4.1. **Seekur Jr Robot Overview**

The Seekur Jr is a four-wheeled skid-steer all-terrain mobile robot produced by Omron Adept – Mobile Robots [23]. It has a built-in computer system that can be used for controlling the robot and interfacing with the onboard sensors [23]. The Seekur Jr in the ISLAB at Memorial University is equipped with a laser rangefinder, depth camera, gyroscope, wheel encoders, Trimble GPS and bumper sensors. Two independent motors drive the wheels on the left and right side of the robot [23]. A dimensioned drawing of the robot is shown in Figure 25.

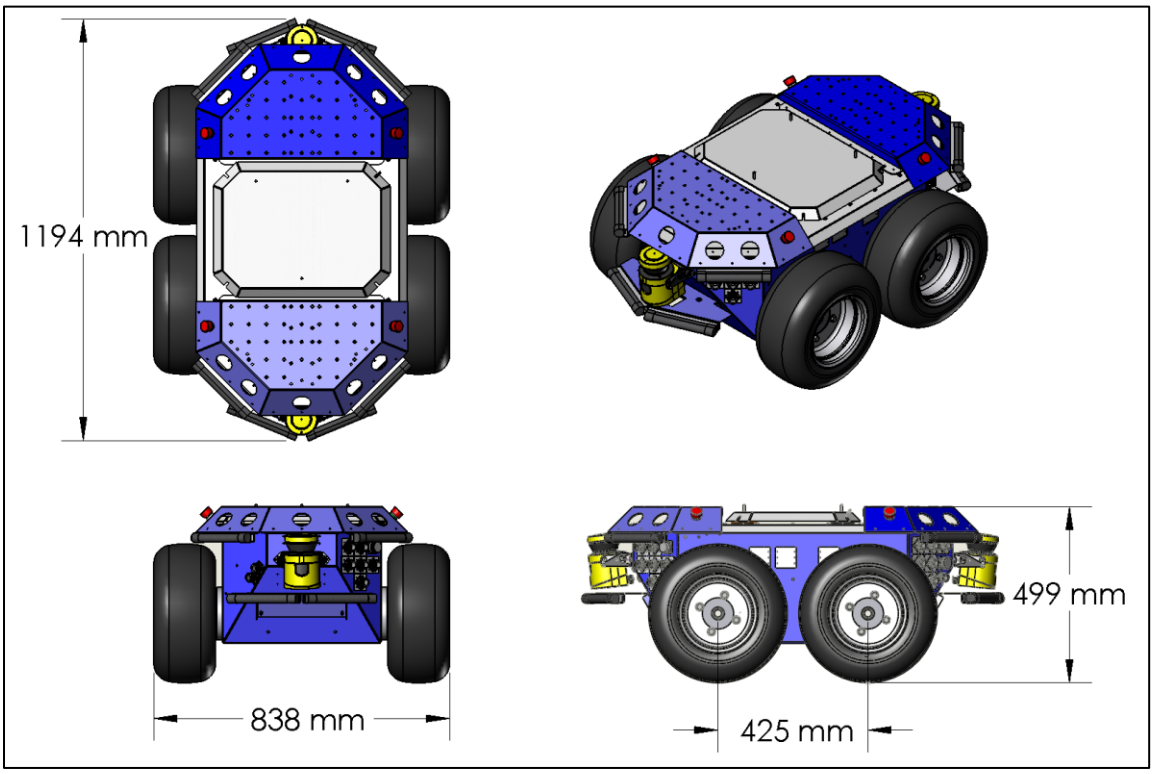


Figure 25 Seekur Jr Robot Physical Dimensions [61]

5.2.4.2. **Advanced Robot Interface for Applications (ARIA)**

The Seekur Jr onboard computer uses the Advanced Robot Interface for Application (ARIA) software developed by Omron Adept – Mobile Robots to manage robot communications, sensing devices and robot internal processes [23]. The onboard computer runs a server application that initializes the selected robot sensors and connects to an ARIA client application [23]. A remote computer runs the client application which forms a Transmission Control Protocol (TCP) connection with the Seekur Jr to establish data and control communications. The client requests specific data packets from the server that contain information such as robot parameters, robot statuses and sensor readings [23]. The client also sends command velocities to the Seekur Jr motor controllers that move the

robot. The client has been integrated with ROS to support data acquisition and modular development of the system.

5.2.4.3. **Robot Operating System (ROS)**

ROS is a platform used for integrating computer software with robotic hardware [68]. The system is comprised of programming libraries and applications specifically designed for robotics research and development. The Seekur Jr experiment primarily uses ROS to gather data from the ARIA client as it is streamed from the onboard ARIA server.

The ARIA client has been configured to publish incoming sensor data as ROS topics, which are data structures that can be accessed by ROS nodes. The data is gathered using the rosbag tool which subscribes to selected ROS topics and stores the data in the time-stamped rosbag format. This process prevents the data from multiple sensors from becoming desynchronized.

5.2.4.4. **Data Acquisition System Configuration**

The Seekur Jr hardware network is shown in Figure 26. The Intel NUC computer [69] is the central data acquisition device that connects to all other devices on the network. ROS and ARIA are both installed on the NUC. The NUC connects to the Seekur Jr via an ethernet connection and through the Seekur Jr Wi-Fi network simultaneously. The NUC runs the ARIA client which connects to the Seekur Jr ARIA server through the ethernet connection. This connection is used to transfer sensor data packets from the Seekur Jr to the NUC and to send control commands from the NUC to the Seekur Jr. The Wi-fi connection is used to initialize the ARIA server/client applications and interface with the

robot during outdoor experiments using a remote computer (laptop). The setup does not have a computer screen; therefore, the laptop is necessary to configure the system when outdoors. The purpose for using three computer devices in this network is to facilitate system development. The NUC computer has many available ports to allow multiple devices to interface with the network and provides a portable platform for testing and developing software.

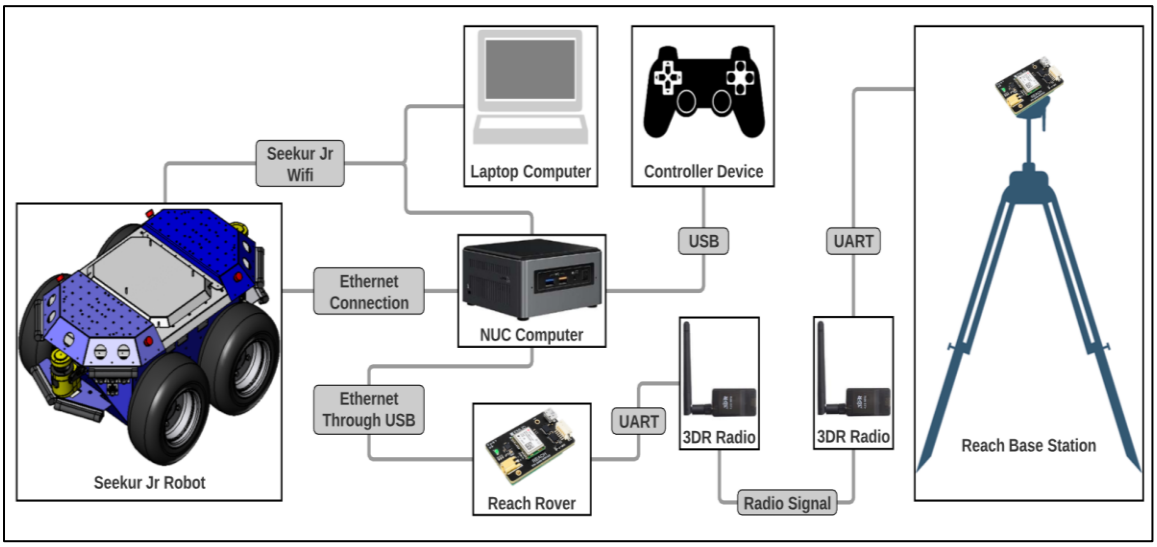


Figure 26 Seekur Jr Network Overview [61] [24] [70] [69]

The Reach rover module streams accelerometer, gyroscope, magnetometer and GPS data through a USB connection to the NUC. The USB connection has been modified to mimic an ethernet connection. A custom TCP server has been installed on the Reach module that preprocesses and sends the Reach IMU data to a custom TCP client ROS node on the NUC. The Reach unit has been configured using the ReachView app to automatically output its GPS data through a specific TCP port (details in Section 5.2.3.2). An additional TCP client ROS node has been created on the NUC to receive the GPS data

through that port. Both client nodes on the NUC parse, timestamp, and publish the sensor data as ROS topics. The data is recorded using the rosbag tool to subscribe to the sensor topics. Figure 27 illustrates the configuration of the NUC with the Reach rover.

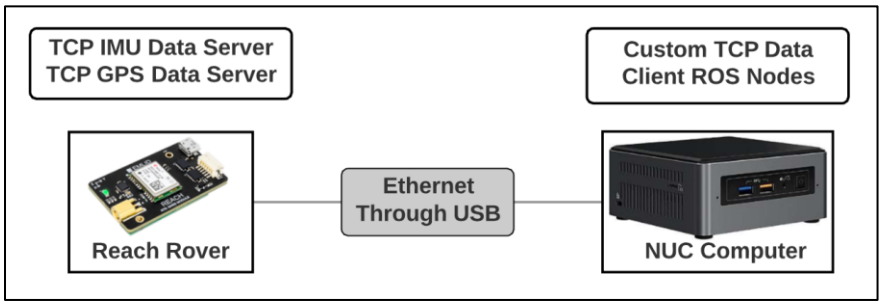


Figure 27 Reach Rover to NUC Connection [24] [69]

The Reach rover receives DGPS corrections and base station parameters from the base station Reach module via a 3DR radio pair. Each Reach unit is connected to a 3DR radio using a six-pin ribbon cable connector. The wiring scheme is shown in Figure 28 below. The data transmits using the RTCM3 format at a baud rate of 57600. The base station Reach unit and 3DR radio are powered by a battery pack using a micro USB cable.

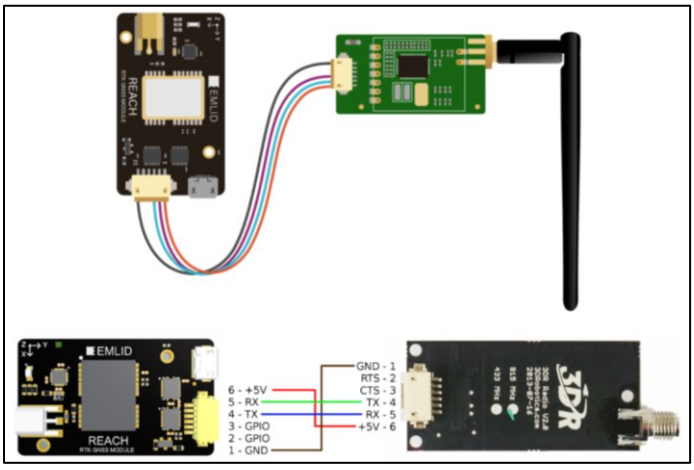


Figure 28 Reach 3DR Radio Wiring Schematic [71]

Using the laptop, the Seekur Jr computer and NUC can be remotely accessed to initialize the ROS data acquisition nodes, the robot controller node and the Reach IMU data server. The laptop connects to the NUC and Seekur Jr through a Secure Shell (SSH) login using their Internet Protocol (IP) addresses to access their root directories. This process is used to initialize the ARIA server on the Seekur Jr during system start-up. During outdoor experiments, the laptop is used to SSH into the NUC root directory. While logged into the NUC, the ARIA client and roscore applications are launched. Using the USB IP address of the Reach rover module, the Reach root directory is accessed from the NUC to initialize the Reach IMU TCP data server. On the NUC, the Reach GPS and IMU data client nodes are launched and the rosbag recorder is activated. A USB gamepad controller is connected to the NUC to control the motion of the robot. The controller uses a ROS node to interface with the ARIA client to send command signals from the NUC to the Seekur Jr onboard computer. This control node is launched while logged into the NUC via the SSH connection with the laptop.

5.2.4.5. **Physical Experimental Setup**

The Seekur Jr experiments were performed on a parking lot behind the Memorial University S. J. Carew Building. The robot was equipped with the data acquisition hardware discussed in section 5.2.4.4 and is shown in Figure 29. The Reach rover GPS antenna was attached to a rigid ground plate to improve signal reception quality [72]. The ground plate was mounted high above the other electronics on the Seekur Jr to reduce potential electronic radio frequency interference. The Reach unit was strapped to the center of the Seekur Jr roof. Figure 30 shows the orientation of the Reach rover on the Seekur Jr.

The sensor axes have been aligned with the robot body frame to simplify coordinate transformations. The 3DR radio receiver for DGPS corrections is strapped to the roof of the Seekur Jr and is connected to the Reach via a universal asynchronous receiver-transmitter (UART) cable. The NUC computer is mounted on the front of the robot. Two 11.1V batteries, wired in series, power the NUC. The Reach rover and 3DR radio are powered through the micro USB connection between the Reach and NUC. The gamepad used to control the robot is connected to the NUC through USB. The wi-fi router for the Seekur Jr is located at the rear of the robot.

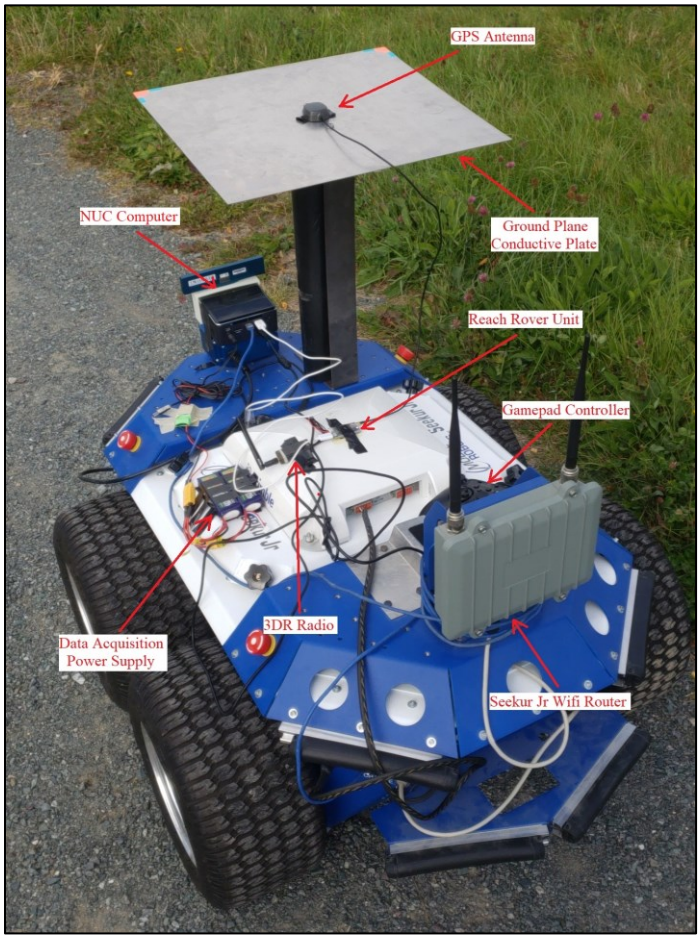


Figure 29 Seekur Jr Data Acquisition System Setup

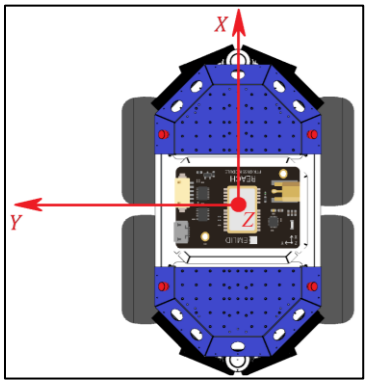


Figure 30 Reach Rover Orientation on Seekur Jr [61] [66]

The DGPS base station is shown in Figure 31. The base station GPS antenna is mounted to a rigid conductive ground plate. The base station reach unit and 3DR radio are securely fastened to the center shaft of the tripod. The 3DR radio is connected to the Reach unit via a UART cable. Both the Reach and radio are powered by the tripod mounted battery.

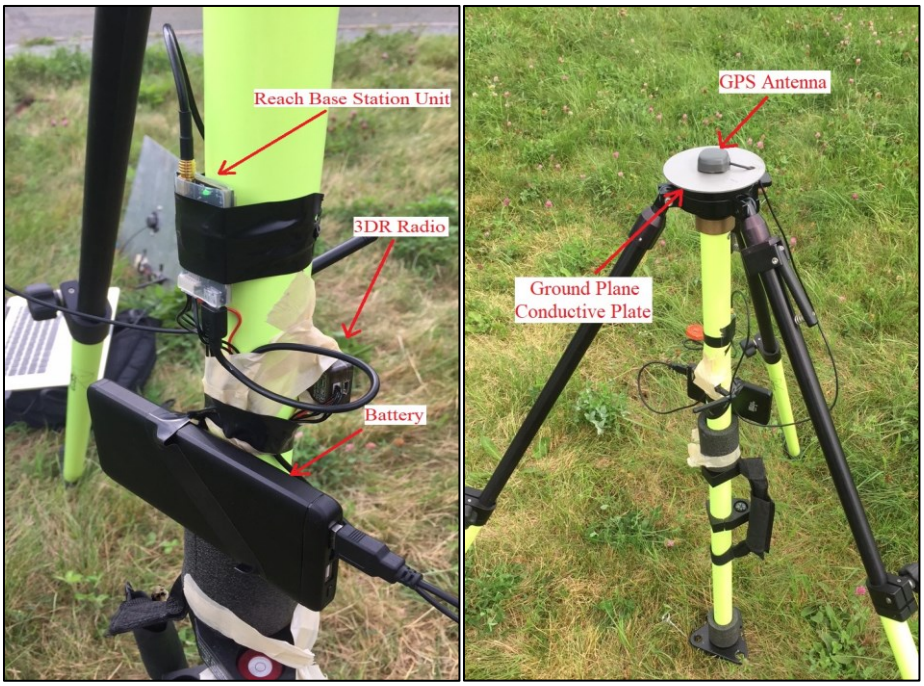


Figure 31 Reach DGPS Base Station Setup

5.2.5. **Sensor Data Processing**

5.2.5.1. **General Data Processing**

During testing, all sensor data was gathered in rosbag format. The data from the Reach IMU was manually parsed using a Python script in ROS. The data from the Reach TCP server was sent in a time-stamped string format containing accelerometer, gyroscope and magnetometer readings in each message. The ROS node receiving the string data deconstructs the string and stores each message in standard ROS data types for IMU data.

The sensor data is post-processed using MATLAB. The built-in rosbag "readMessages" function was used to read the data. The data was manually time synchronized in MATLAB and outputted in matrices for use in the filter experiments. The DGPS measurements were processed using the same procedure discussed in section 4.2.4.

5.2.5.2. **Magnetometer Calibration**

Magnetometers are sensitive to magnetic interference from nearby electrical devices or magnetic objects [73]. This causes magnetometer data to become offset, skewed or scaled which can reduce the quality of INS estimations. Two types of magnetic distortion affect magnetometer performance, hard-iron and soft-iron distortions. Hard-iron distortions are caused by objects that generate a constant magnetic field that is added to magnetometer measurements [73]. Soft-iron distortions are caused by objects that easily become magnetized or demagnetized by magnetic field changes [73].

Simple techniques have been employed to compensate for both sources of distortion in the Reach magnetometer data. The hard-iron distortions are readily removed by applying offsets to the data along each sensor axis. The biases for the magnetometer along each axis are obtained using:

$$\mathbf{m}_{b_i} = \frac{\mathbf{m}_{max_i} + \mathbf{m}_{min_i}}{2} \tag{129}$$

where $\mathbf{m}_b$ is the magnetic bias and i is the sensor axis [74]. The magnetic bias $\mathbf{m}_b$ must be calculated for each individual sensor axis. The bias is subtracted from each data point along each axis.

The soft-iron effects are reduced using a normalized scaling factor. The unnormalized scaling factor is calculated along each axis such that:

$$\mathbf{m}_{s_i} = \frac{\mathbf{m}_{max_i} - \mathbf{m}_{min_i}}{2} \tag{130}$$

where $\mathbf{m}_{s_i}$ is the magnetic scale factor along axis i [74]. The normalized scaling factor is:

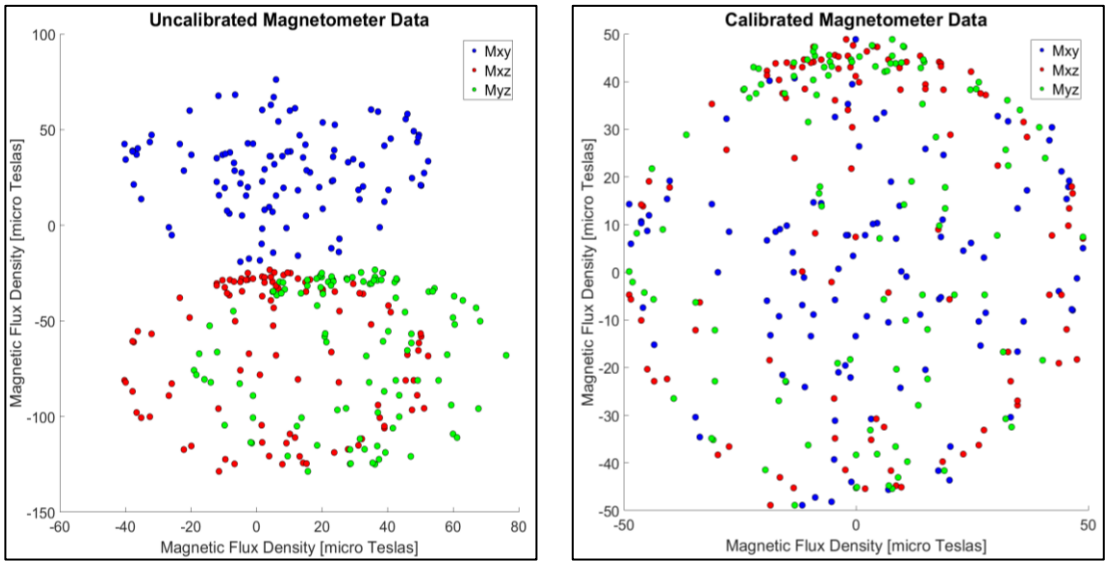
$$\mathbf{m}_S = \frac{\sum_{i=1}^{N} \mathbf{m}_{s_i}}{N} \tag{131}$$

where N is the total number of sensor axes, in this case, three [74]. The calibrated magnetometer data for a given axis is [74]:

$$\mathbf{m}_{c_i} = \mathbf{m}_S\left(\mathbf{m}_i - \mathbf{m}_{b_i}\right) \tag{132}$$

Here, $\mathbf{m}_i$ is the magnetometer data measurement along axis i.

This method was implemented when calibrating the Reach magnetometer for the Seekur Jr experiments. The process was initially applied to a test set of data that was obtained by gathering IMU data while the Reach unit was rotated in "figure-8" patterns along each of the sensor axes. The data in Figure 32(a) shows the uncalibrated magnetometer data which is skewed along each axis, contains offsets relative to the (0,0) origin and is scaled. The calibrated data in Figure 32(b) shows the data has been normalized along each axis. Many of the distortions have been removed and each axis contains symmetrically scaled data values. The data response surface closely resembles a sphere centered on the (0,0) origin.



(a) (b)

Figure 32 Magnetometer Calibration Plots

5.3. Seekur Jr Inertial Navigation Experiment

5.3.1. Inertial Navigation System Filter Implementation

The Seekur Jr INS experiment location is shown in Figure 33. The DGPS trajectory of the robot is plotted in red. The blue circle indicates the starting position of the robot. The yellow circle indicates the location of the DGPS base station during the experiments. The robot path was chosen such that abrupt turning manoeuvres were performed. The DGPS position solution was used as the ground truth for the INS filter. Random Gaussian noise was added to these measurements to mimic the reduced accuracy of a single GPS sensor. The noisy data was used by the INS for robot state estimate tracking.

Figure 33 Seekur Jr Robot Trajectory Experiment Location [75]

The INS algorithm designed in section 5.2.2 was modified to incorporate the Seekur Jr onboard sensors and data processing system. Due to issues encountered with the Seekur Jr encoders, the modified skid-steer ICR-INS model could not be evaluated. The ARIA

client and server were modified to access the Seekur Jr motor data packets, which contain the robot left and right wheel velocities. These velocities can be used to implement the ICR-INS filter, however, when the robot is moving, these data packets stop streaming to the ARIA client. It is likely that this problem is due to Seekur Jr using the requested data packets for control processes, making them unavailable for the client. The modified ARIA client and server were tested using the MobileSim robot simulation software provided by Omron Adept – Mobile Robots [23]. The results show that the motor data packets containing the encoder readings publish correctly from the simulated robot, therefore, the process should work on the Seekur Jr. This issue is to be resolved in future work on this system.

5.3.2. **Seekur Jr Experimental Results and Analysis**

The Seekur Jr INS was tested using both a single EKF and IMM filter for comparison. The graphs in this section represent the results from a single algorithm run using a GPS noise factor of 0.2. The GPS noise factor is a number used during filter testing to control the magnitude of the synthetic noise that was added to the DGPS measurements. A GPS noise factor of 0.2 adds random Gaussian position noise between the values of ± 0.2 meters to the DGPS data to simulate lower GPS accuracy. The filter was tuned using the process discussed in section 4.3.1. The models were tuned using data corrupted by a GPS noise factor of 0.2. The accelerometer, gyroscope and magnetometer data were measured using the Reach module that is attached to the Seekur Jr for navigation purposes. This module uses an MPU-9250 9-axis IMU sensor [76].

The results in Figure 34 show the accuracy of the EKF and IMM filters tracking the robot position in the X-Y plane. The blue line indicates the Reach module DGPS trajectory, which is used as the ground truth in this experiment. The plotted DGPS data does not contain synthetically added noise. The final portion of the robot trajectory (long straightaway segment crossing the S. J. Carew building parking lot in Figure 33) was excluded because the IMM and EKF performed nearly identically for that part of the path. Removing that section yields results that improve the illustrated comparison of an IMM and EKF for trajectories with multiple turning manoeuvres. Note that the mirrored orientation of the robot trajectory in Figure 34 relative to Figure 33 is due to a rotation from the DGPS North-East-Down (NED) frame to the body frame of the Reach module which was not readjusted before plotting. The encircled portions of the graph identify periods where the IMM outperformed the EKF. Near the position $(10\ m, 3\ m)$ at the beginning of the trajectory, the robot was driven down over the edge of a curb. The sudden impact caused the IMU readings to spike. The shock to the measurements was filtered to reduce the stability issues encountered when running the estimators. It can be seen in the encircled area near the impact site, that the IMM filter tracks the robot more accurately than the EKF for a time after the impact. The two other encircled graph segments indicate better tracking by the IMM over the EKF at the beginning of two different turn manoeuvres. Both filters have accurate trajectory tracking for the duration of the test run. The remaining graphs illustrate the results for only the IMM filter for convenience. The filter accurately tracks the position of the robot in all three directions as shown in Figure 35. The curb impact time is approximately $50s$ as shown by the large oscillations in the Z-position in Figure 35.

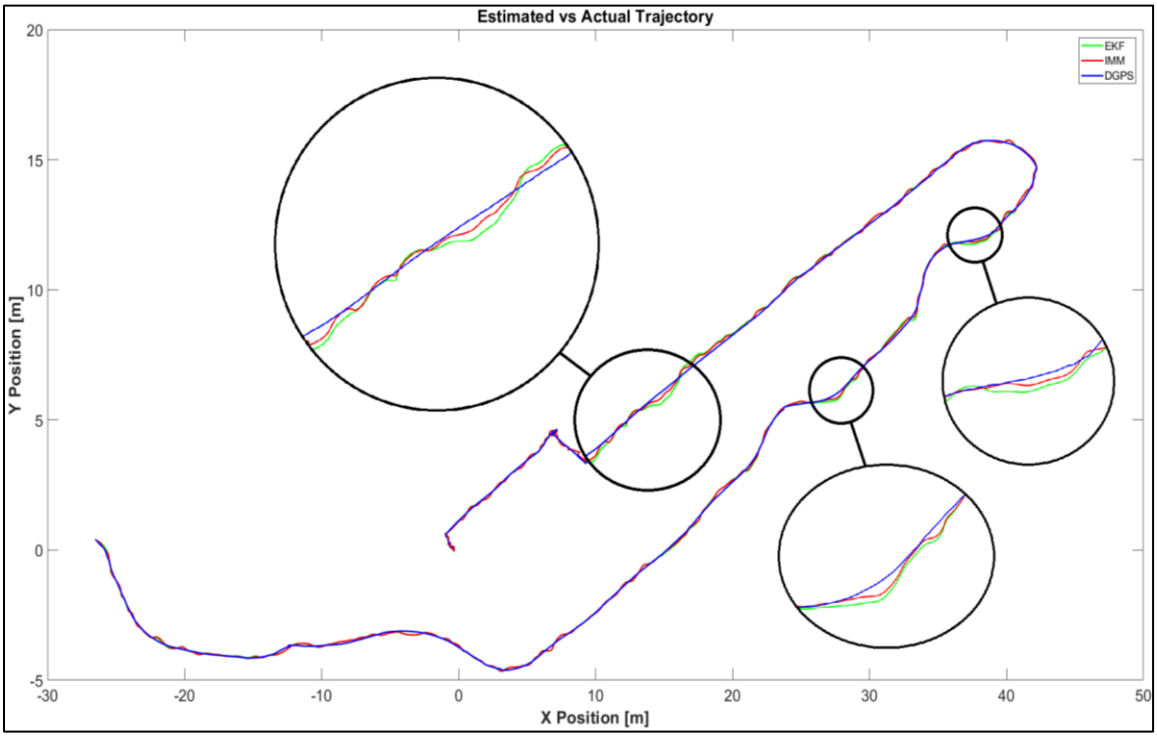


Figure 34 Seekur Jr EKF and IMM-INS Robot Trajectory Estimates

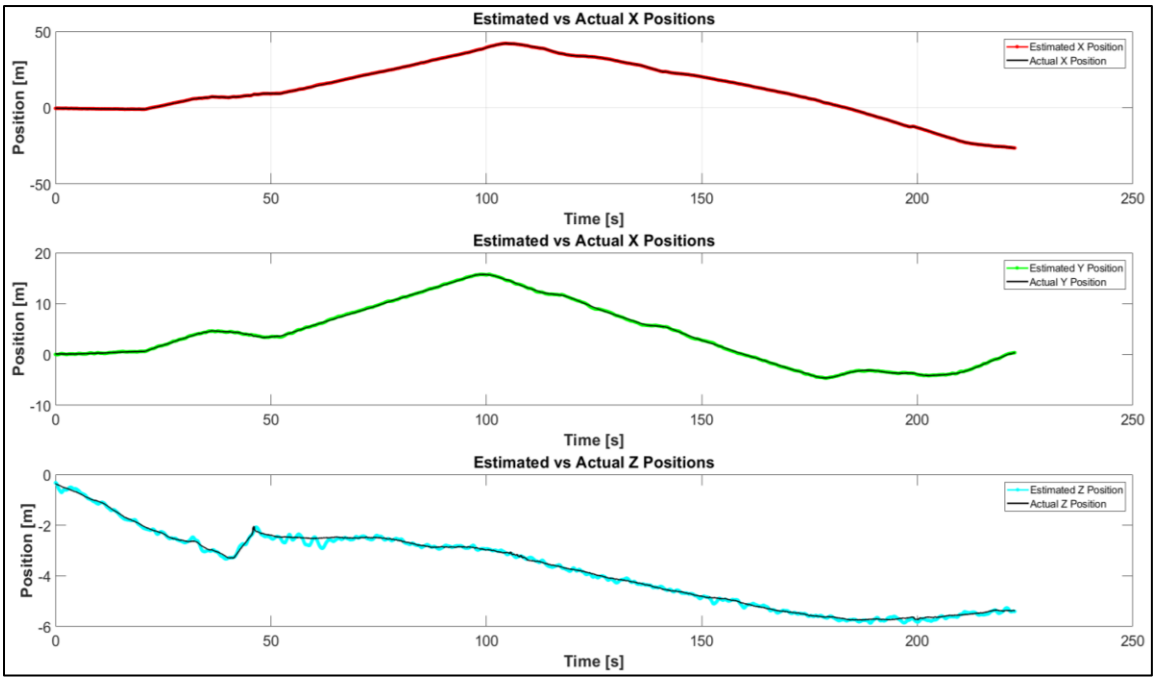


Figure 35 Seekur Jr IMM-INS Position Estimates

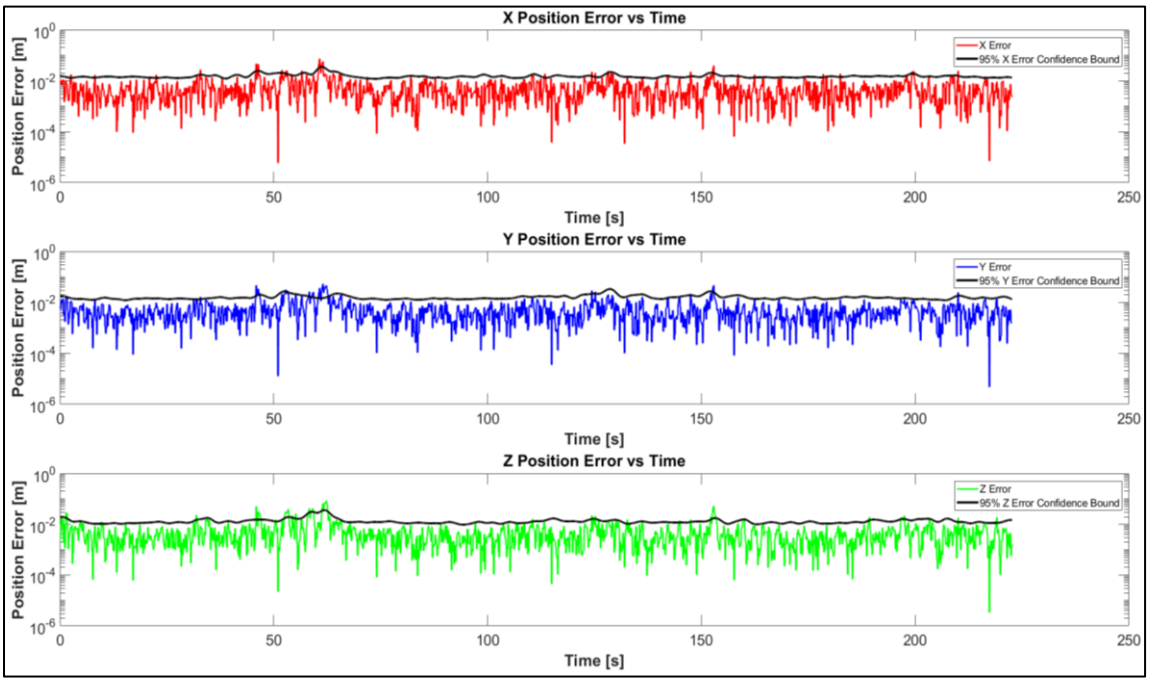


Figure 36 Seekur Jr IMM-INS Position Errors

Figure 36 shows the IMM estimated position errors. The positional errors are generally below by the 95% confidence bounds for each error vector. The errors can be seen exceeding the confidence bounds near times when the robot was performing turns. The results shown in Figure 37 illustrate the differences in the IMM and EKF positional RMS errors. The RMS error magnitude graph in Figure 37 is a measure of the magnitude of the combined RMS error vectors along each axis. The GPS noise factor for each set of trials is on the X-axis of the plots. The trials run the filters 30 times and average the results to more accurately measure filter performance. The averaged results are plotted in Figure 37 and tabulated in Table 7. The error of the IMM filter is lower than the EKF for all tested GPS noise factor values. Based on this observation, it is likely that having multiple models that use different sets of noise parameters can maintain accurate tracking results despite

changing accuracies of sensing devices. This can be especially useful if a sensor experiences interference and the system must switch estimation reliance to other sensors.

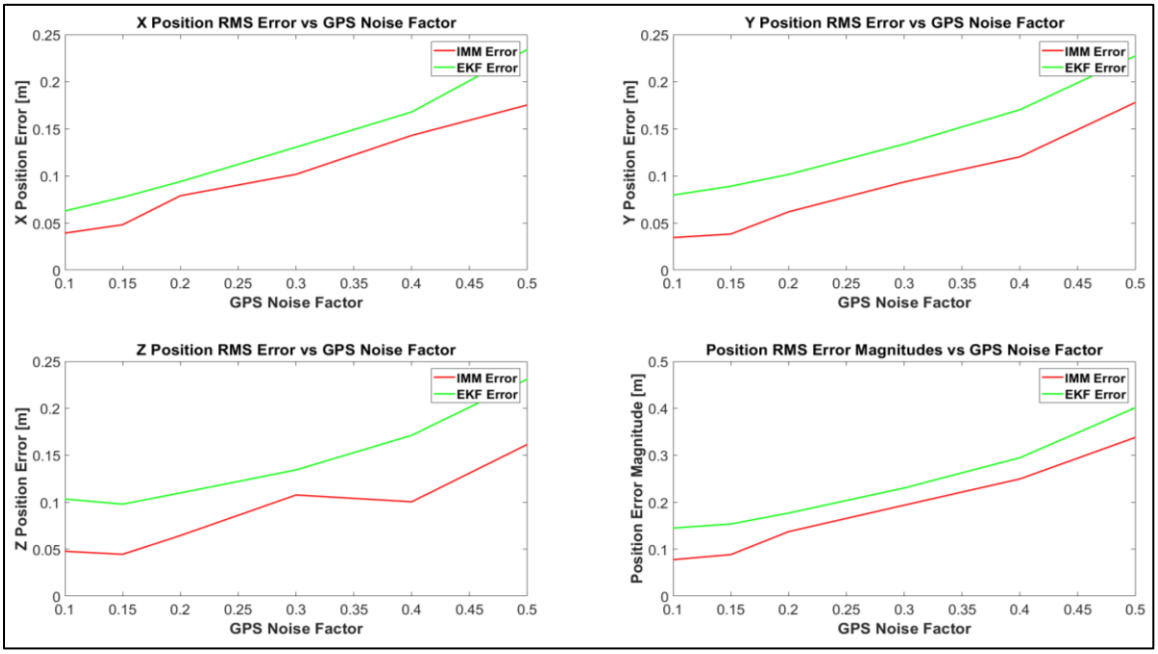


Figure 37 Seekur Jr INS RMS Error Comparison for IMM vs EKF

Table 7 Seekur Jr INS Positional RMS Error Results

Positional RMS Error Results								
GPS Data Noise Factor	**Average X Position RMS Error IMM [m]**	**Average X Position RMS Error EKF [m]**	**Average Y Position RMS Error IMM [m]**	**Average Y Position RMS Error EKF [m]**	**Average Z Position RMS Error IMM [m]**	**Average Z Position RMS Error EKF [m]**	**Average Positional RMS Error IMM [m]**	**Average Positional RMS Error EKF [m]**
0.1	0.0393	0.0628	0.0346	0.0796	0.0478	0.1034	0.0778	0.145
0.15	0.0482	0.0773	0.0383	0.0891	0.0446	0.098	0.0886	0.1537
0.2	0.079	0.094	0.0619	0.1016	0.0646	0.1099	0.1375	0.177
0.3	0.1017	0.1305	0.0936	0.1338	0.1077	0.1344	0.1938	0.2304
0.4	0.143	0.1679	0.1203	0.1703	0.1005	0.1712	0.2496	0.2948
0.5	0.1753	0.2342	0.1781	0.2275	0.1615	0.231	0.3381	0.4013

The velocity estimates of the IMM are shown in Figure 38. The velocity of the robot behaves as expected. The path of the robot trajectory has a slight incline, which is

represented by the low-velocity values in the Z direction. The Y velocity typically has a value of zero, with spikes near turn manoeuvres and the curb drop impact. The spikes are likely due to the skid-steer steering of the robot inducing lateral velocities during turns. The X velocity is the expected profile given the trajectory followed by the robot.

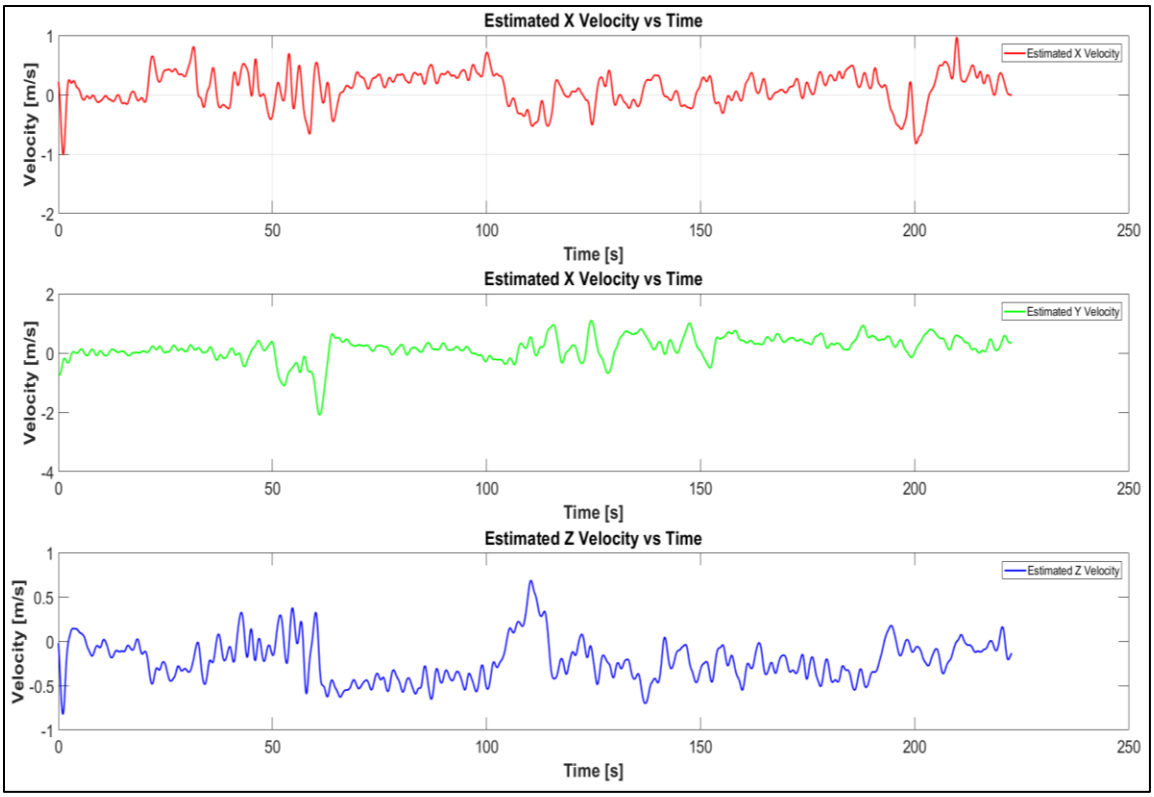


Figure 38 Seekur Jr IMM-INS Filter Velocity Estimates

The roll, pitch and yaw angles estimated by the IMM filter are shown in Figure 39. Near the curb impact time, the orientation angles spike. The robot moved over the curb one wheel at a time, which causes all angles to quickly transition before settling during the first straightaway section. As expected, the roll and pitch of the robot typically remain near constant values during the test run. The estimated changes in the yaw heading match the turns made by the robot.

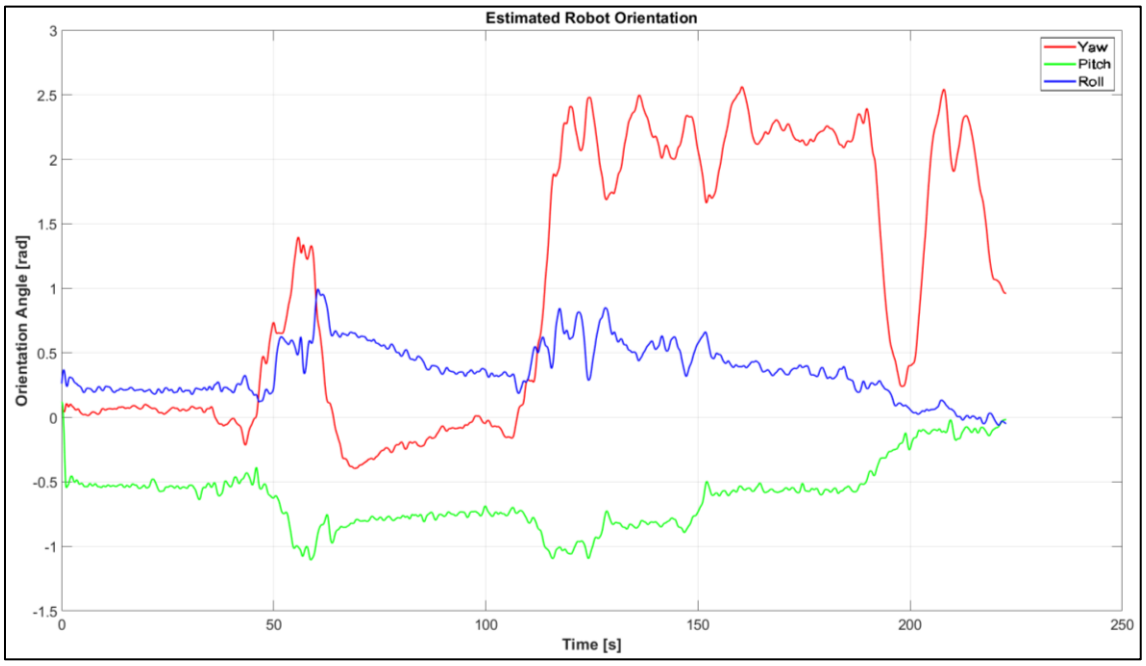


Figure 39 Seekur Jr IMM-INS Roll, Pitch and Yaw Estimates

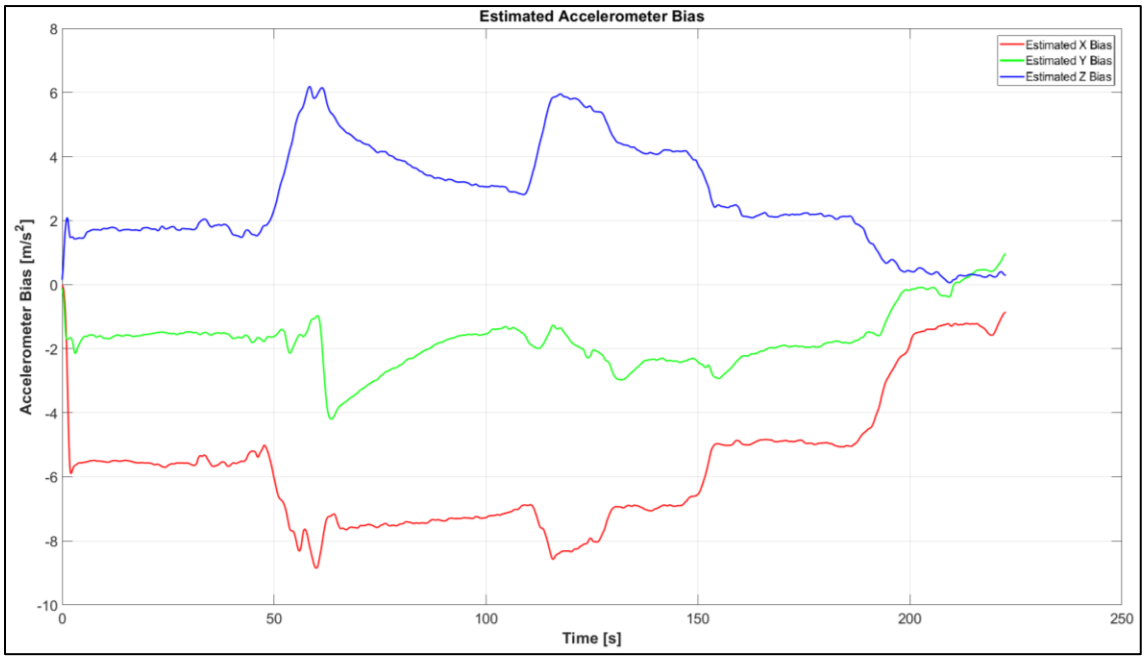


Figure 40 Seekur Jr IMM-INS Accelerometer Bias Estimates

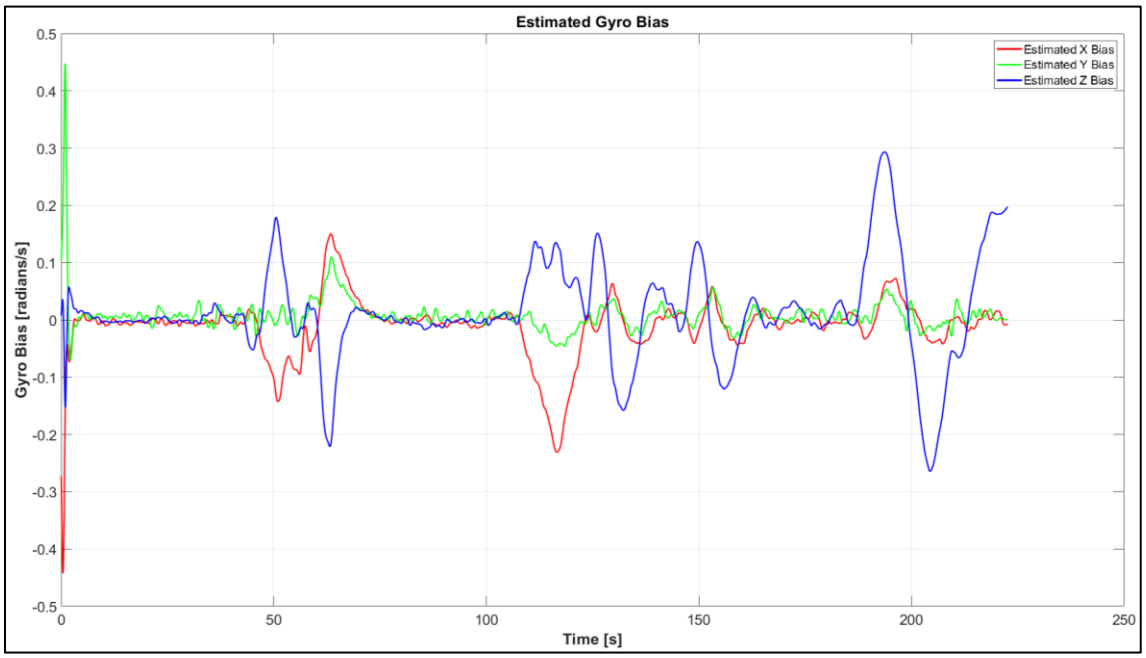


Figure 41 Seekur Jr IMM-INS Gyroscope Bias Estimates

Figure 40 and Figure 41 illustrate the accelerometer and gyroscope biases. In each graph, the biases abruptly change in response to the curb impact and turning manoeuvres. The biases gradually shift towards their steady-state values following each manoeuvre as expected.

The results in Figure 42 indicate the unsmoothed model probabilities for each mode of the IMM-INS. The red line indicates the first mode of the IMM which has the same tuning parameters as the EKF that was tested. This plot shows that this model typically has the higher probability for the duration of the test, however, the influence of the second model is also contributing to the state estimates in the IMM-INS. Near the time associated with the curb impact, it is shown that both models contribute to the combined state estimates approximately evenly. IMM probabilities based on varying noise parameter

models seem to have higher model switching sensitivity than when multiple process models are used as demonstrated in Figure 4. However, this may also be the result of the small differences in the noise tuning parameters of each filter causing both models to rapidly switch due to consistently similar likelihoods.

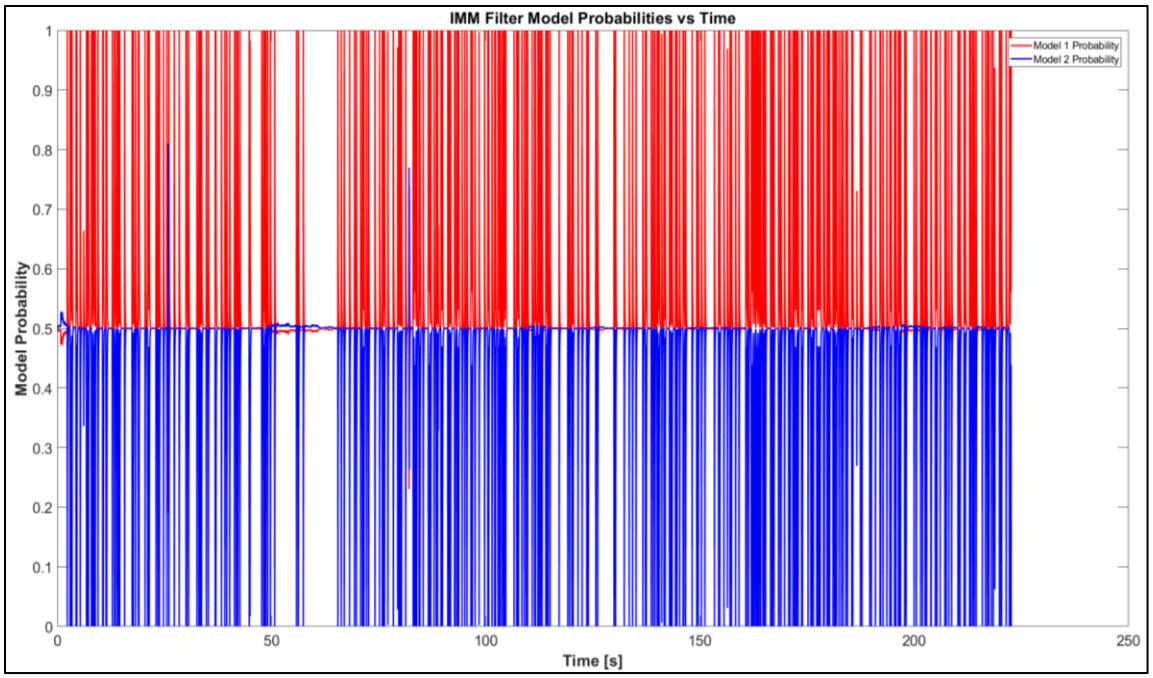


Figure 42 Seekur Jr IMM-INS Model Probabilities

5.4. Conclusions

This chapter has presented a method for implementing skid-steer kinematics in the vehicle state-space model for the INS system and covered the design of a data acquisition system for the Seekur Jr robot to support multi-model estimator research. The performance of the IMM filter has been shown to effectively predict the states of a skid-steer mobile robot. The IMM filter generally outperformed the EKF in terms of RMS positional error.

This result may potentially change depending on the selection of noise parameters. A tuning experiment using a multi-factor design may be required to determine noise parameter interactions and definitively tune the system for future comparisons. The results show that an IMM filter with multiple noise parameter modes can facilitate tuning and achieve good performance without many test trials. The performance differences between the IMM and EKF were most noticeable near the curb impact and turning manoeuvres where the IMM typically maintained better tracking estimates. The multiple modes of the IMM generally reduce the likelihood of the filter failing to track the robot when its dynamic behavior changes. Although using multiple models does not guarantee that the filter will not fail, it does add robustness to the system. If one or more of the IMM models fail to track the system, the likelihood of those models drops to a very low value to remove the influence that those models have on the overall state estimates for the INS.

The experimental testbed designed for the Seekur Jr is a suitable configuration for future IMM localization research. The ARIA/ROS interface and data acquisition system can effectively obtain and process the onboard sensor data. The Emlid Reach modules have been configured for easy use in future experiments. The main concern moving forward with this equipment is the reliability issues experienced with maintaining fixed DGPS carrier lock during outdoor experiments. It is recommended that these modules be replaced in the future with more reliable hardware. Another issue that must be addressed in the future is debugging the encoder data stream issue experienced with ARIA. Obtaining robot side velocities will allow the skid-steer kinematic models to be incorporated in the IMM-INS and tested. It is likely that the modified INS model in section 5.2.2 may improve filter

performance. The recommended course of action for fixing this problem is to identify whether the ARIA server on the Seekur Jr is intercepting the data packets when the robot is moving, and if so, develop code that makes these packets available when the robot is moving.

Chapter 6

Conclusions and Future Work

About this chapter: This chapter discusses the conclusions that were made during the experiments conducted for this thesis. The overall advantages and disadvantages of IMM filtering are discussed with regards to the applications that have been presented. Additional research topics and required work to advance this project further are discussed.

6.1. Conclusions

This thesis has presented an analysis of IMM state estimator performance for computer vision tracking and mobile robot INS localization applications. The following discusses the conclusions for each experiment in terms of the objectives and expected contributions outlined in section 1.3.

6.1.1. Objective 1 Conclusions

Objective 1 – Design an effective computer vision tracking system that implements mean shift and IMM filtering techniques.

The designed computer vision tracker in Chapter 3 effectively tracked targets using mean shift paired with an IMM filter. The colour histogram approach can be inaccurate if the video background and tracked target have similar colours. Changing illumination effects in a video can also cause problems for this method. The vision system also does not

consider any changing scale sizes of tracked objects. The inclusion of an IMM filter assists the tracker by providing information regarding the motion characteristics of the target. Using an IMM filter generally outperformed single motion model Kalman filters in the tested scenarios. The two modes in the IMM allowed the tracker to adaptively switch kinematic models when a target exhibited constant velocity or acceleration behaviour. The benefits of mode switching will likely have the biggest impact in general video target tracking rather than high contrast synthetic videos like those in sections 3.3.1-3.3.4. Additionally, the tracker computational demand can be reduced by adjusting how often the mean shift algorithm is executed. It was observed that the computational time of the IMM filter was significantly less than mean shift operations. Therefore, using the IMM filter to track the target for multiple frames between each mean shift update can potentially improve computing performance.

6.1.2. **Objective 2 Conclusions**

Objective 2 – Demonstrate the effectiveness of IMM filtering for automobile INS applications.

Chapter 4 illustrates the design process for an automobile IMM-INS using a vehicle state-space model. Both the IMM-INS and EKF-INS produced accurate tracking results for the states of the vehicle for the duration of the trajectory. The IMM filter used two modes that contained differently tuned noise parameters for the sensor and process models. The inclusion of multiple noise figures allowed the filter to shift its trust in the onboard sensors adaptively. The positional error of the IMM was typically less than the EKF positional

error. This can be visibly seen in Figure 13 where the IMM maintains better position estimates during two of the turning manoeuvres made by the vehicle.

6.1.3. **Objective 3 Conclusions**

Objective 3 – Design an IMM-INS for skid-steer mobile robots using ICR tracking for outdoor navigation applications.

Chapter 5 provides the framework required for incorporating two-dimensional skid-steer kinematics in the vehicle state-space model. The proposed model should provide improved tracking performance of the Seekur Jr robot when lateral skidding occurs during turns. As shown in [58], the ICR tracking in the EKF can improve filter performance when GPS data become unavailable for periods of time. The evaluated IMM in Chapter 5 demonstrates that the method can outperform an EKF if tuned properly. In all tested scenarios, the IMM filter maintained lower RMS positional error than the EKF. The highlighted trajectory sections in Figure 34 illustrate areas where the IMM visibly performs better than the EKF. These locations correspond to turning manoeuvres and the time after the robot travelled over a curb. These results were expected based on the IMM localization results in Chapter 4.

6.1.4. **Objective 4 Conclusions**

Objective 4 – Develop an experimental testbed for the Seekur Jr robot for multi-model localization research work.

The experimental testbed for the Seekur Jr robot detailed in Chapter 5 is complete except for encoder data streaming from the Seekur Jr motor encoders. The basic functionality of the encoder streaming has been configured and tested using a simulator with successful results. The ROS and ARIA system integration provides the framework for adding and configuring additional sensors for future work with the data acquisition system. The Emlid Reach DGPS equipment has been successfully configured and installed on the Seekur Jr robot. Postprocessing code has been successfully designed for synchronizing, calibrating and exporting Seekur Jr sensor data for state estimator applications.

6.2. **Contributions**

The completion of the objectives in this thesis work has led the following contributions:

Contribution 1 – IMM design and validation for computer vision target tracking and robotic inertial navigation applications. Two strategies for augmenting the model bank of IMM filters (i.e.: models with different process and sensor noise characteristics and models with different system dynamics) were tested and validated. This work was has yielded two publications:

1) P. J. Glavine, O. D. Silva, G. Mann and R. Gosine, "Color-Based Object Tracking using Mean Shift and Interactive Multiple Model Kalman

Filtering," in Newfoundland Electrical and Computer Engineering Conference (NECEC), St. John's, 2017

2) P. J. Glavine, O. D. Silva, G. Mann and R. Gosine, "GPS Integrated Inertial Navigation System Using Interactive Multiple Model Extended Kalman Filtering," in 2018 Moratuwa Engineering Research Conference (MERCon), Moratuwa, 2018

Contribution 2 – Development of an experimental testbed for multiple model estimation based on the Seekur Jr platform. As part of the thesis work the Seekur Jr platform is ROS enabled with access to data streams from 2D Lidar, 3D nodding Lidar, IMU, digital compass, wheel encoder, onboard GPS, RTK DGPS ground truth, and vision sensors.

Contribution 3 – Design and experimental validation of an IMM filter using the developed Seekur Jr mobile robot testbed. The IMM filter strategy was validated for mobile robot navigation purposes in this thesis.

6.3. Future Work

The following discusses potential research directions for the work completed in this thesis:

- Computer Vision Tracker – Quantitative analysis of the mean shift IMM filter performance for general video tracking should be tested to determine tracker effectiveness for real-world video applications. Scale-invariant target

representation using image characteristics other than colour histogram should be explored to address the scaling and variable illumination issues. The reduction of the computational demand caused by using an IMM with the mean shift algorithm should also be quantitatively determined in the future.

- Automobile INS – Further analysis of the effects of multiple noise models can be explored for this system. Expanding the number of modes used by the IMM may improve tracking accuracy results. The INS should be tested with a dataset from a vehicle that has performed many aggressive turning manoeuvres during its trajectory. Adding a dynamic model to the IMM that includes the Ackermann steering configuration of a typical automobile would also likely improve estimator performance.
- Skid-steer Robot INS – The skid-steer kinematics discussed in Chapter 5 should be implemented in a future version of the Seekur Jr INS. Additionally, skid-steer models that include lateral wheel forces and wheel-ground interactions should be considered if the INS is tested on skid-steer robots or vehicles that have high velocities during operation. In [58], it was found that some of the assumptions employed in their ICR tracking method may breakdown for high-velocity skid-steer systems.
- Seekur Jr Testbed – The encoder issue with the Seekur Jr should be further investigated to make the data available for future use in multi-model filter experiments. The Emlid Reach DGPS system should be eventually replaced with more reliable and user-friendly hardware. This recommendation results from the

difficulty of maintaining DGPS fixed position solutions during field tests, and the abundance of difficulties experienced when initially configuring the modules for use in the data acquisition system.

- Indoor/Outdoor IMM – The IMM-INSs in this thesis mainly focused on outdoor navigation where GPS is available. Future work with these filters can include developing multiple models that rely on different sensors for either indoor or outdoor navigation. The IMM filter would likely facilitate the mode switching of the INS when the system transitions between outdoor and indoor environments. Furthermore, within the indoor/outdoor IMM modes, there can be additional modes implemented that contain multiple sets of noise parameters for shifting trust between sensors when operating in unstructured environments.

Bibliography

[1] V. Villani, F. Pini, F. Leali and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics,* vol. 55, pp. 248-266, 2018.

[2] P. Kemao, D. Miabo, C. B. M., C. Guowei, L. K. Yew and L. T. H., "Design and Implementation of a Fully Autonomous Flight," in *26th Chinese Control Conference*, 2007.

[3] M. W. Powell, T. Crockett, J. M. Fox, J. C. Joswig, J. S. Norris, K. J. Rabe, M. McCurdy and G. Pyrzak, "Targeting and Localization for Mars Rover Operations," [Online]. Available: https://www-robotics.jpl.nasa.gov/publications/Mark_Powell/IEEE_IRI_06_Targeting.pdf. [Accessed 24 November 2015].

[4] O. Kanoun and H.-R. Tränkler, "Sensor Technology Advances and Future Trends," *IEEE Transactions on Instrumentation and Measurement,* vol. 53, no. 6, pp. 1497-1501, 2004.

[5] N. Sundaram, "Making Computer Vision Computationally Efficient," Berkeley, 2012.

[6] J. A. Farrell, Aided Navigation GPS with High Rate Sensors, The McGraw-Hill Companies, 2008.

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

[8] R. E. Kalman, "A New Approach to Linear Filtering," *Transactions of the ASME–Journal of Basic Engineering,* pp. 35-45, 1960.

[9] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI*, Orlando, FL, United States, 1997.

[10] C. Stachniss, "SLAM Course - 06 - Unscented Kalman Filter (2013/2014; Cyrill Stachniss)," Youtube, 2013.

[11] P. Abbeel, "EKF, UKF," University of California, Berkeley, [Online]. Available: https://people.eecs.berkeley.edu/~pabbeel/cs287-fa13/slides/EKF_UKF.pdf.

[12] M. Xin and S. N. Balakrishnana, "A New State Observer and Flight Control of Highly Maneuverable Aircraft," in *American Control Conference*, St. Louis, MO, USA, 2009.

[13] C. B. Low and D. Wang, "GPS-Based Path Following Control for a Car-Like Wheeled Mobile Robot With Skidding and Slipping," *IEEE Transactions on Control Systems Technology,* vol. 16, no. 2, pp. 340-347, 2008.

[14] A. F. Genovese, "The Interacting Multiple Model Algorithm for Accurate State Estimation of Maneuvering Targets," *Johns Hopkins APL Technical Digest,* vol. 22, no. 4, pp. 614-623, 2001.

[15] X. R. Li and Y. Bar-Shalom, "Design of Interacting Multiple Model Algorithm for Air Traffic Control Tracking," *IEEE Transactions on Control Systems Technology,* vol. 1, no. 3, pp. 186-194, 1993.

[16] R. Radhakrishnan, A. K. Singh, S. Bhaumik and N. K. Tomar, "IMM-Cubature Quadrature Kalman Filter for Manoeuvring Target Tracking," *IEEE International*

Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2015.

[17] T. Cho, C. Lee and S. Choi, "Multi-Sensor Fusion with Interacting Multiple Model Filter for Improved Aircraft Position Accuracy," *Sensors,* vol. 13, pp. 4122-4137, 2013.

[18] D. Svensson and L. Svensson, "A New Multiple Model Filter With Switch Time Conditions," *IEEE Transactions on Signal Processing,* vol. 58, no. 1, pp. 11-25, 2010.

[19] K. Jo, K. Chu and M. Sunwoo, "Interacting Multiple Model Filter-Based Sensor Fusion of GPS With In-Vehicle Sensors for Real-Time Vehicle Positioning," *IEEE Transactions on Intelligent Transportation Systems,* vol. 13, no. 1, pp. 329-343, 2012.

[20] R. Toledo-Moreo, M. A. Zamora-Izquierdo and B. Úbeda-Miñarro, "High-Integrity IMM-EKF-Based Road Vehicle Navigation With Low-Cost GPS/SBAS/INS," *IEEE Transactions on Intelligent Transportation Systems,* vol. 8, no. 3, pp. 491-511, 2007.

[21] O. Törő, T. Bécsi, S. Aradi and P. Gáspár, "Sensitivity and Performance Evaluation of Multiple-Model State Estimation Algorithms for Autonomous Vehicle Functions," *Journal of Advanced Transportation,* vol. 2019, 2019.

[22] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," *International Journal of Robotics Research (IJRR),* 2013.

[23] Omron Adept Mobile Robots, Seekur Jr Manual, 2017.

[24] Emlid, "Reach RTK docs: Introduction," [Online]. Available: https://docs.emlid.com/reach/img/reach/Reach_400x400-400x380.png. [Accessed 2019].

[25] F. Dellaert, D. Fox, W. Burgard and S. Thrun, "Monte Carlo Localization for Mobile Robots," in *IEEE International Conference on Robotics & Automation* , Detroit, Michigan, 1999.

[26] J.-S. Gutmann, W. Burgard, D. Fox and K. Konolige, "An Experimental Comparison of Localization Methods," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Victoria, B.C., Canada, 1998.

[27] J. E. Guivant and E. M. Nebot, "Optimization of the Simultaneous Localization and Map-Building Algorithm for Real-Time Implementation," *IEEE Transactions on Robotics and Automation,* vol. 17, no. 3, pp. 242-257, 2001.

[28] A. J. Davison, I. D. Reid, N. D. Molton and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 29, no. 6, pp. 1052-1067, 2007.

[29] J. W. Marck, A. Mohamoud, E. v. Houwen and R. v. Hejster, "Indoor Radar SLAM a Radar Application for Vision and GPS Denied Environments," in *European Radar Conference*, Nuremberg, Germany, 2013.

[30] M. F. Fallon, J. Folkesson, H. McClelland and J. J. Leonard, "SLAM, Relocating Underwater Features Autonomously Using Sonar-Based," *IEEE Journal of Oceanic Engineering,* vol. 38, no. 3, pp. 500-513, 2013.

[31] P. Biber and W. Strasser, "The Normal Distributions Transform: A New Approach to Laser Scan Matching," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 2003.

[32] H. Zhang, L. Li and W. Xie, "Constrained Multiple Model Particle Filtering for Bearings-Only Maneuvering Target Tracking," *IEEE Access,* vol. 6, pp. 51721-51734, 2018.

[33] H. Qian, D. An and Q. Xia, "IMM-UKF Based Land-Vehicle Navigation With Low-Cost GPS/INS," in *IEEE International Conference on Information and Automation*, Harbin, China, 2010.

[34] G. Pandey, J. McBride and R. Eustice, "Ford Campus Vision and Lidar Data Set," *International Journal of Robotics Research,* vol. 30, no. 13, pp. 1543-1552, 2011.

[35] K. Y. K. Leung, Y. Halpern, T. D. Barfoot and H. H. T. Liu, "The UTIAS Multi-robot Cooperative Localization and Mapping Dataset," *International Journal of Robotics Research,* vol. 30, no. 8, pp. 969-974, 2011.

[36] O. D. Silva, G. K. I. Mann and R. G. Gosine, "An Ultrasonic and Vision-Based Relative Positioning Sensor for Multirobot Localization," *IEEE Sensors Journal,* vol. 15, no. 3, pp. 1716-1726, 2015.

[37] T. R. Wanasinghe, G. K. I. Mann and R. G. Gosine, "Decentralized Cooperative Localization Approach for Autonomous Multirobot Systems," *Journal of Robotics,* 2016.

[38] K. Ogata, Modern Control Engineering Fifth Edition, Prentice Hall, 2010.

[39] S. Rezaei and R. Sengupta, "Kalman Filter-Based Integration of DGPS and Vehicle Sensors for Localization," *IEEE Transactions on Control Systems Technology,* vol. 15, no. 6, 2007.

[40] E. L. Haseltine and J. B. Rawlings, "Critical Evaluation of Extended Kalman Filtering and Moving-Horizon Estimation," *Industrial & Engineering Chemistry Research,* vol. 44, pp. 2451-2460, 2005.

[41] A. Tsalatsanis, K. Valavanis and A. Yalcin, "Vision Based Target Tracking and Collision Avoidance for Mobile Robots," *Journal of Intelligent & Robotic Systems,* vol. 48, no. 2, 2007.

[42] B. Maurin, O. Masoud and N. P. Papanikolopoulos, "Tracking All Traffic: Computer Vision Algorithms for Monitoring Vehicles, Individuals, and Crowds," *IEEE Robotics & Automation Magazine,* pp. 29-36, March 2005.

[43] A. Cesetti, E. Frontoni, A. Mancini, P. Zingaretti and S. Longhi, "A Vision-Based Guidance System for UAV Navigation and Safe Landing using Natural Landmarks," *Journal of Intelligent and Robotic Systems,* vol. 57, no. 1-4, 2010.

[44] D. Comaniciu, V. Ramesh and P. Meer, "Kernel-Based Object Tracking," [Online]. Available: http://comaniciu.net/Papers/KernelTracking.pdf.

[45] E. Karami, M. Shehata and A. Smith, "Segmentation and Tracking of Inferior Vena Cava in Ultrasound Images using a Novel Polar Active Contour Algorithm," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Montreal, QC, Canada, 2017.

[46] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework," *International Journal of Computer Vision,* vol. 56, no. 3, pp. 221-255, 2004.

[47] D. Comaniciu, V. Ramesh and P. Meer, "Real-Time Tracking of Non-Rigid Objects using Mean Shift," *Proceedings IEEE Conference on Computer Vision and Pattern Recognition,* 2000.

[48] M. Shah, "Lecture 11: Mean Shift," UCF Center for Research in Computer Vision, 2012.

[49] R. Collins, "Mean Shift Tracking CSE598G Spring 2006," [Online]. Available: http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf.

[50] A. Bhattacharyya, "On a Measure of Divergence between Two Multinomial Populations," *The Indian Journal of Statistics,* vol. 7, no. 4, pp. 401-406, 1946.

[51] N. Duinker, "Tutorial - Learn How To Juggle 3 Balls," 2013. [Online]. Available: https://www.youtube.com/watch?v=x2_j6kMg1co.

[52] M. Barczyk and A. F. Lynch, "Invariant Observer Design for a Helicopter UAV Aided Inertial Navigation System," *IEEE Transactions on Control Systems Technology,* vol. 21, no. 3, pp. 791-806, 2013.

[53] J. Solà, "Quaternion kinematics for the error-state Kalman filter," 2017. [Online]. Available: http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf.

[54] M. T. Mason, "Lecture 8. Quaternions," 2013. [Online]. Available: http://www.cs.cmu.edu/afs/cs/academic/class/16741-s07/www/lectures/Lecture8.pdf.

[55] Solutions, Oxford Technical, *RTv2 GNSS-Aided Inertial Measurement Systems User Manual,* 2015.

[56] N. Trawny, X. S. Zhou, K. Zhou and S. I. Roumeliotis, "Inter-robot Transformations in 3D," *IEEE Transactions on Robotics,* vol. 26, no. 2, pp. 226-243, 2010.

[57] R. Hermann and A. Krener, "Nonlinear Controllability and Observability," *IEEE Transactions on Automatic Control,* Vols. AC-22, no. 5, pp. 728-740, 1977.

[58] Google, "Google Maps," 2017. [Online]. Available: https://www.google.ca/maps/@49.0534004,8.3962994,245m/data=!3m1!1e3.

[59] J. Pentzer and S. Brennan, "Model-based Prediction of Skid-steer Robot Kinematics Using Online Estimation of Track Instantaneous Centers of Rotation," *Journal of Field Robotics,* pp. 455-476, 2014.

[60] T. Wang, Y. Wu, J. Liang, C. Han, J. Chen and Q. Zhao, "Analysis and Experimental Kinematics of a Skid-Steering Wheeled Robot Based on a Laser Scanner Sensor," *Sensors,* vol. 15, pp. 9681-9702, 2015.

[61] W. Yu, O. Y. Chuy, E. G. Collins Jr. and P. Hollis, "Analysis and Experimental Verification for Dynamic Modeling of A Skid-Steered Wheeled Vehicle," *IEEE Transactions on Robotics,* vol. 26, no. 2, pp. 340-353, 2010.

[62] Omron Adept Mobile Robots, *Seekur Jr Technical Drawings,* Omron Adept Mobile Robots.

[63] Trimble, "All About GPS: Error Correction," [Online]. Available: https://www.trimble.com/gps_tutorial/howgps-error.aspx. [Accessed 2019].

[64] NovAtel, "An Introduction to GNSS Chapter 4 GNSS Error Sources," [Online]. Available: https://www.novatel.com/an-introduction-to-gnss/chapter-4-gnss-error-sources/error-sources/. [Accessed 2019].

[65] Emlid, "How RTK Works," [Online]. Available: https://docs.emlid.com/reach/common/tutorials/rtk-introduction/. [Accessed 2018].

[66] T. Takasu, RTKLIB ver. 2.4.2 Manual, RTKLIB, 2013.

[67] Emlid, "Reach RTK docs: Mechanical specs," [Online]. Available: https://docs.emlid.com/reach/img/reach/specs/reach-dimensions.png. [Accessed 2019].

[68] Emlid, "Reach RTK docs: Introduction," [Online]. Available: https://docs.emlid.com/reach/common/reachview/img/reachview/introduction/reachview.gif. [Accessed 2019].

[69] Open Robotics, "About ROS," [Online]. Available: https://www.ros.org/about-ros/. [Accessed 2019].

[70] Scan, "Intel Dual Core 8th Gen i3 Tall NUC Barebone Mini PC Kit," [Online]. Available: https://www.scan.co.uk/images/products/super/2972672-l-a.jpg.

[71] 3D Robotics, 3DR Radio V2 Quick Start Guide.

[72] Emlid, "Reach RTK docs: Hardware Integration," [Online]. Available: https://docs.emlid.com/reach/hardware-integration/. [Accessed 2019].

[73] Emlid, "Reach RTK docs: Quick Start," [Online]. Available: https://docs.emlid.com/reach/quickstart/. [Accessed 2018].

[74] J. E. Lenz, "A Review of Magnetic Sensors," *Proceedings of the IEEE,* vol. 78, no. 6, pp. 973-989, 1990.

[75] K. Winer, "Simple and Effective Magnetometer Calibration," [Online]. Available: https://github.com/kriswiner/MPU6050/wiki/Simple-and-Effective-Magnetometer-Calibration.

[76] Google, "Google Maps," 2019. [Online]. Available: https://www.google.ca/maps/@47.5747935,-52.7366979,272m/data=!3m1!1e3.

[77] InvenSense, "MPU-9250 Product Specification Revision 1.1," 20 June 2016. [Online]. Available: http://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf. [Accessed 2019].