Online Autonomous Extrinsic Calibration of an Inertial Measurement Unit using Gaussian Radial Basis Function Neural Networks

by

Gregory Mifflin

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 5, 2023

Keywords: IMU, extrinsic calibration, system identification, neural network

Abstract

This thesis presents a fully online and autonomous method to extrinsically calibrate an inertial measurement unit (IMU) to the body frame of a vehicle. Extrinsic sensor calibration is an important step in obtaining valid information in the vehicle frame, without which an autonomous vehicle cannot function properly. Traditionally, a manual calibration routine must be performed by a set of trained experts to a high degree of precision before the vehicle can be safely operated. This procedure costs time and money and limits the design of the sensor suite. An online and autonomous calibration method would eliminate this constraint, saving time, and allowing for the dynamic reconfiguration of the sensor suite. The autonomous IMU-to-Vehicle calibration procedure presented in this thesis is conducted in a two-stage process. First, a gaussian radial basis function neural network is used to emulate the output of a virtual IMU in the vehicle frame. Then, a maximum likelihood search algorithm estimates the extrinsic calibration parameters by performing an IMU-to-IMU calibration between the IMU on the body of the vehicle, and the emulated IMU. The IMU emulation method obtains high-fidelity acceleration estimates on both simulated and experimental data sets. The maximum likelihood search method obtains sensor position estimates within 2 mm of the true sensor location in each direction and within 0.2 degrees of the true sensor orientation for a battery of tests in simulation. In experimental tests, this method estimated the true lateral and longitudinal sensor positions to within 3 cm, and the true sensor orientation to within 0.25 $\circ$ in each direction.

Acknowledgments

This thesis and its contributions would not have been possible without the help of many different people. I would like to thank my advisor, Dr. Bevly, for providing direction to my work, especially in those times when I was unsure how to approach a specific research problem. Howard Chen also deserves acknowledgment for his support of my work. In the early phase of my graduate research, Howard was invaluable for his willingness to give considered and insightful criticism that helped me refine my approach to the problem, and for always taking an interest in my success as an engineer.

I would also like to thank John David for each and every time he helped troubleshoot and resolve a software issue that I had. Without his help, I would have had far more difficulty implementing the software developed for this thesis. It should also be said that John David's work in reconfiguring the hardware setup on the MKZ made data collection much easier and saved me many days of frustration at the track.

Thank you to Stephanie as well, for helping me capture the ground truth data and collect experimental data. The laser surveyor is difficult to use alone, and your help in finding the marked points and reading off the measured values made it substantially easier to collect accurate ground truth positions and orientations. I also appreciated your help in troubleshooting the hardware configuration and driver software every time I collected data at the track, and your help in thinking through the difficult problem of aligning measurements from different survey positions. Thank you for helping me collect the data, even though the repeated figure-8's made you nauseous.

I am grateful to Luke and Dan for their generosity, in giving advice, and in providing a room for me to stay in between leases. I won't forget the hospitality you gave me in a difficult situation.

I am also grateful for the support of my parents during these years I have spent as a graduate student, and for all the years prior. You have constantly pushed me to aim high, and excel in every endeavor. I would not be where I am today if you had not instilled in me a fascination with the natural world, and a drive to solve challenging problems.

Finally, I want to thank Tyler for helping me with class projects and building every sort of mechanical and electrical device that I am incapable of making myself, despite my degree. Your enthusiasm for exploring the boundaries of science and technology, and some admittedly wild ideas, have inspired me since the day I arrived in Auburn. I'm sure we'll have plenty more adventures in the days to come.

Table of Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

ADC	Analog-to-Digital Conversion
$ARIMA$	Auto-Regressive Integrated Moving Average
$ARMA$	Auto-Regressive Moving Average
ARX	Auto-Regressive with Exogenous Variables
CG	Center of Gravity
CMA	Covariance Matrix Adaptation evolutionary strategy
CP	Control Point
$DARPA$	Defense Advanced Research Projects Agency
DVL	Doppler Velocity Log
$ECEF$	Earth-Centered Earth-Fixed reference frame
ECI	Earth-Centered Inertial reference frame
EKF	Extended Kalman Filter
FOG	Fiber-Optic Gyroscope
$GNSS$	Global Navigation Satellite System
GPS	Global Positioning System
$GRBFNN$	Gaussian Radial Basis Function Neural Network

GT	Ground Truth
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IV	Instrumental Variable
$LiDAR$	Light Detection and Ranging
LM	Levenberg-Marquardt optimization algorithm
$LSTM$	Long Short-Term Memory
$MEMS$	Micro Electro-Mechanical Systems
MLP	Multi-Layer Perceptron
$MSDP$	Multi-State Dependent Parameters
NN	Neural Network
ORB	Oriented Fast and Rotated BRIEF
$RAKF$	Regularized Adaptive Kalman Filter
$RBFNN$	Radial Basis Function Neural Network
RGB	Red-Green-Blue three number color model
RIV	Refined Instrumental Variable
RLG	Ring-Laser Gyroscope
RLS	Recursive Least Squares
RNN	Recurrent-Neural Network
RPE	Recursive Prediction Error

SDP	State-Dependent Parameters
$SURF$	Speeded-Up Robust Features
$TDOA$	Time Difference of Arrival
TVP	Time-Varying Parameters
UKF	Unscented Kalman Filter

Reference Frames

c	Camera frame
i	Inertial frame
n	Navigation frame
r	Road frame
s	Sensor frame
v	Vehicle frame
w	World frame

Variables

$\Delta\vec{a}_t$	Acceleration estimation error at time t
Δu_t	Shortest distance to the training dataset for test sample at time t
ϵ	Error
η	Noise vector
λ	Exponential forgetting factor
Φ	Wavelet

ϕ	system state model parameters
ψ	Noise parameter vector
ρ	Parameter vector
θ	error state model parameters
$\vec{\alpha}_{jk}^{k}$	Angular acceleration of frame k with respect to frame j, resolved in frame k
$\vec{\omega}_{jk}^{k}$	Angular velocity of reference frame k with respect to frame j, resolved in frame k
$\vec{\psi}_{jk}$	Euler angle misalignment between reference frames j and k
$\vec{a}_{jk}^{k}$	Acceleration of reference frame k with respect to frame j, resolved in frame k
$\vec{r}_{jk}$	Lever arm between reference frames j and k
$\vec{v}_{ij}$	measurement of point i in frame j
$\vec{v}_{jk}^{k}$	Velocity of reference frame k with respect to frame j, resolved in frame k
A	State transition matrix
B	Input coefficient matrix
b_{f_g}	Bias for quantity f_g
C	Output matrix
C_t	State coefficient matrix
e_i	I^{th} basis vector
$Hist$	History of inputs
J	Sum of squared errors
L	Lag operator

R_j^k	Rotation matrix from reference frame j to reference frame k
u	Exogenous input vector
U_{train}	Set of all input vectors in the training dataset
w	Weight multiplying a basis function
x_k	K^{th} input
X_t	System state vector at time t
y_k	K^{th} output
z_t	instrumental variable vector
I	Identity Matrix
T	Transformation between two reference frames

Chapter 1

Introduction and Motivation

An autonomous vehicle is defined as any physical system capable of maneuvering from a given start location to a defined end location without human aid [28]. This broad definition does not specify the environment in which the vehicle is required to operate, nor the method of locomotion. Hence a wide range of different systems can be adequately termed "autonomous vehicles"; this can include not only self-driving cars, but also drones, submarines, and even legged mobile robots. Regardless of the physical platform, or the environment that it operates in, four basic tasks need to be undertaken in order to achieve the navigation goal: perception, localization, planning and control [28, 37]. Perception is the task upon which the success of the other three rests. Without a proper understanding of the external environment, and the internal state of the vehicle, localization becomes more difficult. Without reliable perception and localization, planning a path through the environment becomes impossible, and without a path, there is nothing to direct control of the physical system.

Operating autonomously in the world requires the system to make decisions about how to control its actuators safely and effectively [60, 37, 51]. Information about the surrounding environment and the vehicle's state is necessary to make the correct decisions. Autonomous vehicles rely on a set of sensors to gather information about their external environment or about their internal state. The physical measurements from these sensors are fed into the vehicle's software stack, where the data is refined, fused and analyzed to obtain a model of the world state, and the vehicle state at any given time [37, 43]. Hence it is crucial that the sensors are chosen and placed to provide sufficient and accurate information about the world state [57]. A sensor suite for an autonomous vehicle commonly consists of a set of inertial measurement units, cameras, radars, LiDARs, odometers, and GPS antennas. Intrinsic and extrinsic calibration is required in order for this information to be valid for the vehicle's frame of reference. A manual calibration routine must be performed before the vehicle leaves the factory to ensure that it can properly operate in an autonomous mode.

1.1 Manual Calibration

In order to minimize systematic error, a rigorous and time-consuming calibration process must be undertaken when the sensors are mounted on the vehicle, and every time the sensor configuration is changed [56]. A sample of this calibration process is illustrated in Figure 1.1. Sensors cannot be reconfigured as needed by the vehicle operator, and if a sensor becomes misaligned or misplaced through real-world events, another expensive manual calibration routine needs to be performed to correct the issue [91]. These manual calibration routines typically involve a number of stages. IMUs are first intrinsically calibrated; this entails estimating the bias and noise properties of both the accelerometer and the gyroscope for each device. This is done by measuring the Allan variance across time while the IMU is stationary. Each IMU must also be extrinsically calibrated. The misalignment between the vehicle frame and the sensor frame is determined manually after the sensor has been fixed to a mounting structure at a specific reference location. First, an IMU measurement is obtained at the reference location and the IMU is coupled to each of the mounting structures in a defined sequence. While the IMU is coupled to all of the mounting structures, a new set of measurements is taken for each mount. For each reference location, the misalignment angles are calculated.

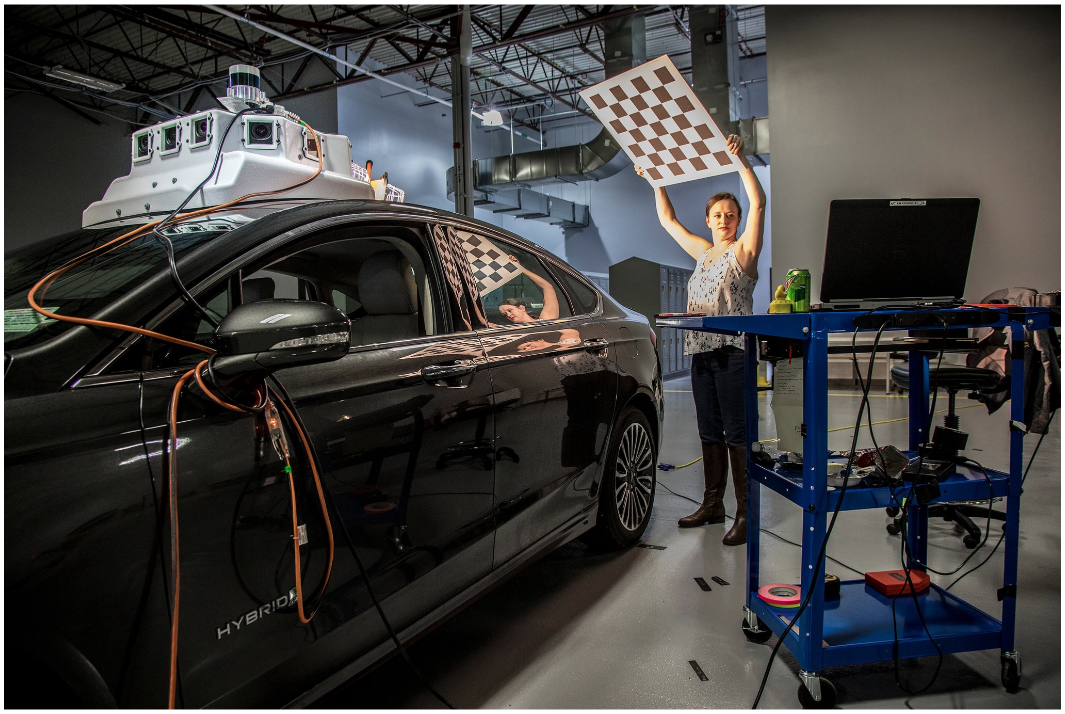


Figure 1.1: A manual calibration takes place in front of a calibration target [73].

The factory calibration procedure involves many steps, several of which require a trained professional to perform them [56] [91]. The procedure needs to be performed any time the sensors are knocked out of alignment. Without training to perform the manual calibration, the driver would need to bring their vehicle in to be re-calibrated each time this happens. Additionally, the manual calibration assumes a fixed placement of sensors. This means that the sensors cannot be reconfigured online as desired. An efficient, online, and autonomous calibration technique would remove these barriers. In order to do this, a calibration from the sensors relative to the vehicle frame must first be developed. However, direct Sensor-to-Vehicle calibration has yet to be thoroughly addressed in the literature [5].

1.2 Background

Autonomy has always been a central objective in the field of robotics [39]. Automata have existed in some form at least since the 18th century, when Vaucanson, a prodigal French engineer, constructed several marvelous mechanical contraptions, touring them around Europe for an exorbitant ticket price. However, they would remain objects of amusement for another two centuries [29, 60]. It wasn't until the middle of the 20th century that the economic potential of automation was recognized. Autonomous robotic manipulators were first developed to improve the efficiency of industrial manufacturing processes. However, the technology needed to realize autonomy did not yet exist. Robots were limited to drive-by-wire and master-slave systems.

The rapid development of the computer in the post-war period enabled robots to be programmed electronically for the first time [60]. This led to the development of assembly line robots, independently by Devol and Engelberger, that were pre-programmed to follow a specific routine. In precisely controlled situations, these repetitive actions were sufficient. The need for sensory feedback, however, became increasingly clear by the 1960's. Haptic feedback had already been employed by master-slave systems, but visual input was more difficult to employ. Algorithmic techniques, known since as classical computer vision, allowed a crude form of object recognition [77]. In 1968, McCarthy used these new computer vision techniques to develop a robot that could "see" blocks on a table, maneuver over them, and manipulate them with its robotic arm [71].

It was at this point that a mobile robot, capable of navigating its environment to reach a set goal, became plausible [39]. The first step towards this project was made by Inoue, who created a robot that used classical artificial intelligence methods coupled with visual and LiDAR object detection to perform initial positioning for an assembly task [55, 60, 77]. At the same time, Nevins investigated sensing techniques for manipulating objects based on compliance [60, 78]. New navigation techniques, such as the A* algorithm, were introduced in the 1970's, taking advantage of the nascent exteroceptive sensing capabilities to plan paths and avoid obstacles on the way to a predetermined goal [37]. Early autonomous robot's such as the Stanford Research Institute's "Shakey" pictured in Figure 1.2, could now plan paths through complex environments. Once a viable path had been planned, navigation was merely a matter of control, and this field was already deep enough by the late twentieth century to confidently handle the task. The perception piece was generally performed by calculating the 3D positions of features out of a stereo camera.

While visual navigation works well in static and highly structured environments, it tends to fail in low-structured or dynamic environments [77]. This can lead to catastrophic collisions when the vision system maneuvers too close to a particular obstacle, but also in open spaces. The navigation of these primitive mobile robots was also limited by errors in the visual self-positioning model. If there are too few matching features between the current frame and the previous frame, then the 3D rotation and translation solver might not converge to the correct transform, leading to drastic positioning errors. These issues prevented any serious attempt to create a road-driving autonomous ground vehicle for the time being. The AI winter that took hold in the 1980's did not relent until at least the mid 1990's. By this time, however, a new technology had made positioning a much simpler problem, and interest in autonomous vehicles was reignited.

Figure 1.2: 'Shakey' was an early attempt at an autonomous mobile robot [19].

The Global Positioning System (GPS) offered an objective, global positioning solution that could be compared to road maps obtained from global geographic surveys [1]. Once a path was planned in world frame coordinates, navigation on the open road could be reduced to a path-following problem [39]. While obstacle detection and avoidance were still issues that needed to be addressed, they were no longer globally important, insofar as a vehicle could now feasibly return to a predetermined path and reach its goal. Multi-path and atmospheric effects still hampered GPS solutions, but integration with inertial navigation systems through Kalman filtering enabled a combined positioning solution that was generally accurate to within a few meters save in the most challenging of environments [16].

The latent potential for a surface road capable self-driving car culminated in the DARPA grand challenge in 2004 [1, 39, 72]. The challenge demanded that teams complete a 150 mile course through the Mojave desert. As shown in Figure 1.3, the course consisted of a combination of interstate and more physically demanding off-road terrain. The rough terrain and several hairpin turns proved to be the undoing of all of the competitors. Most vehicles either experienced mechanical problems, became trapped on rocks or steep hills, or flipped over after attempting a tight turn. No vehicle completed more than 7.4 miles of the course. Despite the perceived failure of the event, it resulted in a wave of enthusiasm for self-driving [52, 72]. This renewed interest ultimately led to increased investment in the field, and a redoubling of efforts to complete a race course autonomously [39, 100]. This was achieved only a year later, by no less than five different teams. The success demonstrated the feasibility of autonomous ground vehicles with existing technology, leading to the development of a commercial interest in the field.

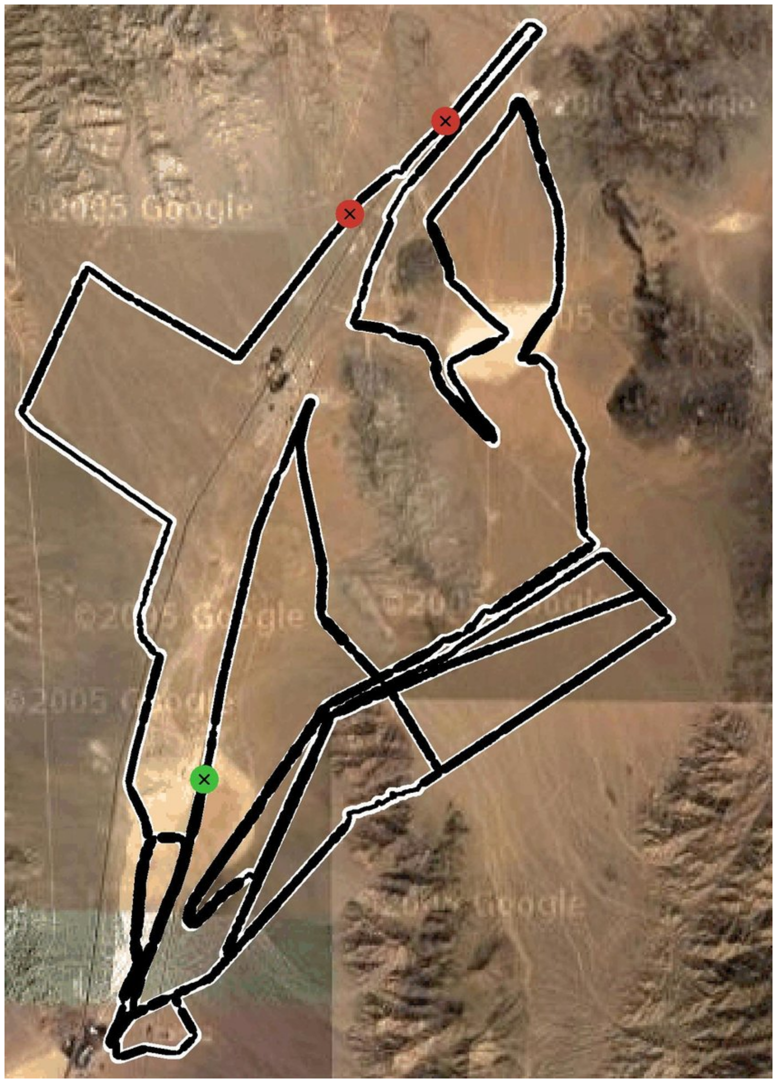


Figure 1.3: The DARPA Grand Challenge map [75].

Since the DARPA grand challenge, focus has shifted away from navigating in rough terrain and towards increasing the autonomy of the system in highly dynamic environments, such as urban centers [39, 100]. This change can be explained by the birth of new companies seeking to monetize the concept, either for personal operation or as part of a taxi fleet. The important challenge for these new investment sources is to operate smoothly in the world's largest cities, where heavy traffic, pedestrians, and urban canyons are a fact of life. To ensure safe performance even in the most challenging circumstances, modern self-driving cars employ extensive sensor suites, and run redundant object detection, localization, and path planning algorithms in parallel [43, 60]. Redundancy ensures that the system as a whole will continue to perform as desired, even when one component of the software architecture fails [37]. However, it also requires a great deal of computational resources, as well as coupling between sensors at both a low and a high level.

1.3 Inertial Measurement Unit

An inertial measurement unit (IMU) is a device that contains sensors that measure both linear accelerations and angular velocities in three dimensions [25, 26]. The IMU is the basic unit employed in inertial navigation. By double-integrating the linear accelerations, an estimate of the relative position change from the initial time to the current time can be obtained, and by integrating the angular rates, an estimate of heading can be determined as well [106]. Hence a single IMU can be used to obtain an estimate of the full 6-D pose of the vehicle, without any need for an external reference. This 'dead-reckoned' position solution, while easy to obtain, degrades severely over time due to a combination of inherent noise and bias in the sensor's measurements. The drift in the position solution can range from centimeters to kilometers over time, making standalone inertial navigation infeasible for surface-road capable autonomous vehicles [53, 106]. Several varieties of IMU exist, each of which acquire the relevant measurements in different ways.

Autonomous vehicles have advanced in capability in recent years in large measure due to the wide suite of sensors mounted on them, and a set of advanced sensor fusion algorithms that enable the system to control the vehicle and make important real-time decisions, such as braking and turning, using their diverse array of measurements [46]. However, in order to obtain useful information from those measurements, the sensors need to be properly calibrated first [53]. Calibration error can result in drift in the position solution, relative localization error, and misperception of the external environment. The effect on the navigation and control of the vehicle is deleterious. A high degree of systematic error in attitude determination, global positioning, or relative positioning breaks the control algorithm's ability to direct the vehicle along a desired path [28]. Hence, precise calibration is crucial to the success of any autonomous vehicle.

1.4 Sensor-to-Sensor Extrinsic Calibration

Extrinsic calibration is the process of estimating the orientation and lever arm between a sensor's reference frame, and another specified reference frame. The orientation and lever arm constitute the rotational and translational components of the pose between the two reference frames, which can then be used to transform measurements between them. The measurements can then be processed in the navigation algorithm's native frame, which is generally the vehicle frame. Hence, extrinsic calibration is an important step in processing information obtained by sensors mounted on an autonomous vehicle.

While direct Sensor-to-Vehicle calibration remains an undeveloped field, Sensor-to-Sensor calibration techniques are well-known. Autonomous calibration procedures have been developed to estimate the relative pose of numerous pairs of sensors. For any two sensors ordinarily used on an autonomous vehicle, a calibration routine exists that can determine their relative pose. In particular, reliable calibration techniques have been developed between an IMU and an array of other sensors [15, 36, 62, 88].

1.4.1 IMU-to-IMU Calibration

IMU-to-IMU extrinsic calibration is a relatively simple procedure, and in most cases can be performed using a least mean squares algorithm given manageable noise and bias in the measurements [54, 59, 62, 88]. A multi-IMU system is considered to have a fixed pivot sensor and N sensors that need to be calibrated to the pivot. This involves determining the N different poses of the IMUs on the autonomous platform defined as a set of coordinate frame transformations:

$$T_0 ... T_N \tag{1.1}$$

which each include a rotation matrix and a translation vector, as shown below.

$$T_k = [r_k, R(\psi_k)] \tag{1.2}$$

The translation vector is composed of the relative position in each of the three dimensions of cartesian space resolved in the target reference frame. The estimated misalignment vector is composed of three euler angles that rotate the k^{th} frame into the target frame.

$$\vec{r}_k = \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} \tag{1.3}$$

$$\vec{\psi}_k = \begin{bmatrix} \theta_k \\ \phi_k \\ \psi_k \end{bmatrix} \tag{1.4}$$

These coordinate frame transformations are estimated by minimizing the error between each IMU's measurements and the measurements in the base frame transformed into the nth coordinate frame T_n across an arbitrary time series. Different loss functions can be used to measure the error, but the Mahalanobis distance, a variant of mean squared error weighted by a covariance matrix, is typically used:

$$T_n^* = min_{T_n}[\sum_{n=1}^{N}\sum_{t=1}^{T}||res(\alpha_t^{T_n}, \hat{\alpha}_t^{T_n})||_\Sigma^2] \tag{1.5}$$

$$\left\|\vec{x} - \vec{\mu}\right\|_\Sigma^2 = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1}(\vec{x} - \vec{\mu})} \tag{1.6}$$

where T_n^* is the optimal pose estimate for the nth IMU, and res() gives the residual between the measurements in the base frame transformed by T_n and those measured by the n^{th} IMU. This transformation of the IMU measurements is accomplished using the rigid body assumption. Since deformation of the vehicle is negligible, it can be assumed that it is a rigid body, and hence, the angular velocity is the same at any point on the body of the vehicle. This invariance allows the following geometric relationship to be derived:

$$\hat{a}_{ik,t}^0 = (R_k^0)^{-1}[\hat{a}_{ik,t}^0 + \left[\hat{\omega}_{ik,t}^0\right]_\times^2 \vec{r}_{k0} + \left[\hat{\alpha}_{ik,t}^0\right]_\times \vec{r}_{k0}] \tag{1.7}$$

where $|\cdot|_\times$ is the skew matrix form of a vector, used to compute cross products. This relationship allows $\hat{a}_{ik,t}^0$ to be calculated from $\hat{a}_{ik,t}^0$ and T_k. Hence the relative pose between two IMUs on the body of a vehicle can be estimated.

1.5 IMU-to-Vehicle Extrinsic Calibration

IMU-to-Vehicle extrinsic calibration involves the determination of the lever arm and misalignment angles between an IMU and the vehicle frame. The primary difficulty in performing direct Sensor-to-Vehicle calibration is the lack of a reference from which to gather the information necessary to obtain a relative pose estimate [5, 22, 67, 74]. Sensor-to-Sensor calibration is generally performed by extracting information about the transformation between the sensor frames that is indirectly observable from the measurements of each. The linear accelerations and angular rates recorded by an IMU, for example, are affected in a deterministic way by its orientation relative to a fixed reference frame [107]. The lever arm between IMUs at different locations on a rigid body is observable in the difference between their measurements, because the rigid-body assumption relates the accelerations at any point on a rigid body's surface to any other point on its surface.

The vehicle itself, on the other hand, does not record any measurements, and hence cannot provide any information, direct or indirect, about its pose relative to a given sensor. This vital information must therefore be gathered from another source. However, for the online calibration task, the available information is limited to the sensors on board the vehicle. The problem as presented is intractable. To solve it, either some external reference is needed to provide additional information about the vehicle's pose, or some way of extracting global information about the vehicle from the sensor suite, which could then be used to indirectly observe the vehicle's pose.

A paper published by J. Almazan in 2013 estimates the yaw misalignment between the vehicle, and a smartphone placed inside it, using the smartphone's embedded IMU and GPS [5]. The yaw angle between the smartphone and the vehicle frame is estimated by assuming perfectly longitudinal acceleration along a straight section of road. On straight sections of road, [5] reports a typical yaw misalignment error of approximately 1-2 % of the total misalignment angle. However, the calibration routine also requires that the vehicle be driven in a straight line, which limits the practical applicability of the algorithm.

A similar approach was developed by Yuanxin Wu in *Versatile Land Navigation using Inertial Sensors and Odometry, In-Motion Alignment and Positioning.* [107]. In contrast to [5], this method estimates both the IMU-to-Vehicle misalignment and lever arm, using a combination of inertial measurements and wheel odometry, without the need for measurements in the world frame in the form of GPS position fixes. It relies on the non-holonomic constraint, which the assumes that the linear acceleration in the vehicle frame is perfectly longitudinal. As long as the assumptions made in the paper, including the non-holonomic constraint, are kept, this method can provide an accurate estimate of the pitch and yaw misalignment angles, as well as an estimate of the lever arm. However, the authors noted that this method can take an extremely long time to converge to a solution, and requires a high-grade IMU.

Some prior research has attempted to solve the IMU-to-Vehicle extrinsic calibration problem without the using the non-holonomic constraint [67]. Instead, a single-track dynamic model, and a model of the vehicle suspension, are used to approximate the linear accelerations in the vehicle frame [38, 90]. With these vehicle models and a recursive least squares (RLS) estimator, this method is capable of obtaining the 3D misalignment between the sensor frame and the vehicle frame without a prior Vehicle-to-World frame calibration [35, 67]. The technique makes use of both wheel odometry and GPS position fixes to aid the extrinsic calibration process, and takes the slope of the road surface into consideration as well. Whereas the method developed by Wu is only viable for a high-grade IMU, the method presented in [67] works for an automotive-grade IMU. However, the method does not estimate the IMU-to-Vehicle lever arm, and requires a dynamic model to be hand-tuned for each vehicle that it is deployed on.

While much effort has been spent on developing cutting edge Sensor-to-Sensor calibration techniques, direct Sensor-to-Vehicle calibration has yet to be thoroughly discussed in the literature. The prior work covered here represents the most relevant among the several methods that exist in the literature to perform direct IMU-to-Vehicle calibration. These methods generally rely either on vehicle motion constraints or a dynamic vehicle model. Vehicle motion constraints are only valid for a limited range of possible calibration maneuvers, which limits the feasibility of the methods that use them. Alternatively, it has been shown that a dynamic vehicle model can replace vehicle motion constraints in the extrinsic calibration process, without restricting the calibration maneuver [67]. However, building an accurate vehicle model is a task of its own that requires special attention. Many methods exist to determine a model for a physical system from data, and from this wide variety of potential solutions, the best candidate needs to be selected. This process falls under the domain of system identification.

1.6 Contributions

The contributions of this thesis consist of the development and experimental validation of a two-stage IMU-to-Vehicle extrinsic calibration method. This method can be applied autonomously while a vehicle is online, without any need for manual intervention. It could then be used to enable autonomous extrinsic calibration of the entire sensor suite of an autonomous ground vehicle. An itemized list of the contributions is given below:

1. A neural network dynamic vehicle model, which emulates the output of an IMU at a specified location on the vehicle using only global vehicle states

2. A custom maximum likelihood search method, which estimates the position and orientation of a physical IMU relative to a virtual IMU emulated in the vehicle frame

3. The combination of the dynamic model and maximum likelihood search into a two-stage IMU-to-Vehicle extrinsic calibration method

4. Experimental validation of the developed extrinsic calibration method against precise ground truth data through a series of tests on a real-world platform

1.7 Thesis Outline

Chapter 1 has introduced the IMU-to-Vehicle extrinsic calibration problem, and motivated its solution as a way to increase autonomy and eliminate the expenses associated with manual calibration. Chapter 2 surveys the system identification techniques that were considered for the IMU emulation phase of the calibration technique developed in this thesis. Chapter 3 explains why the gaussian radial basis function neural network was selected from among the considered system identification techniques, and discusses the implementation of this model in detail. Chapter 4 covers the custom implementation of maximum likelihood search that was used to estimate the extrinsic calibration parameters given the IMU model developed in the previous chapter. Chapter 5 details the training and test procedures of the extrinsic calibration method, so that it can be implemented on a real-world platform. Chapter 6 provides the results of the IMU emulation and pose estimation phases on a battery of tests in simulation. The simulated results validate the calibration method, and give a rough upper bound on its performance. Chapter 7 presents all of the experimental results collected for this thesis. The IMU emulation and pose estimation results are analyzed for a number of different calibration maneuvers. Chapter 7 also includes a discussion of the experimental setup and collection of ground truth sensor positions and orientations, an overview of the filtering applied to the raw IMU measurements, a test of the sensitivity of the pose estimation process to error in the IMU model, and an analysis of the extrinsic calibration performance as a function of the calibration time. Finally, chapter 8 states the conclusions of this thesis, and provides direction for future work on Sensor-to-Vehicle extrinsic calibration.

Chapter 2
System Identification

System identification is the domain of mechanical and electrical engineering concerned with the specification of the dynamic model of a physical system. This can be accomplished either by derivation using known physical laws, or by empirical analysis. In this thesis, system identification will be useful in constructing a model that can accurately emulate the each component of acceleration in the body frame of a ground vehicle.

It has been demonstrated that an accurate model of the vehicle's dynamics provides enough information to perform a direct IMU-to-Vehicle calibration [67]. However, the vehicle model must be accurate in order to be useful in any calibration scheme, since an error in the reconstructed linear accelerations would generate error in the pose estimate of the sensor. The classical method of modelling physical systems involves a geometric model of the system, and assumptions about the magnitude and direction of various forces on the system based on known physical laws [104]. A wide variety of physical models exist for ground vehicles in particular, ranging from basic kinematic models to the complex and highly parametric nonlinear models that are used in high fidelity simulation environments such as CarSim [7]. The model that is used most commonly in industry and academia is the "Bicycle Model" [83]. The free-body diagram for the bicycle model is shown in Figure 2.1. It shows how the vehicle accelerations are derived from the forces applied by the ground surface to the wheels through tire-road forces.

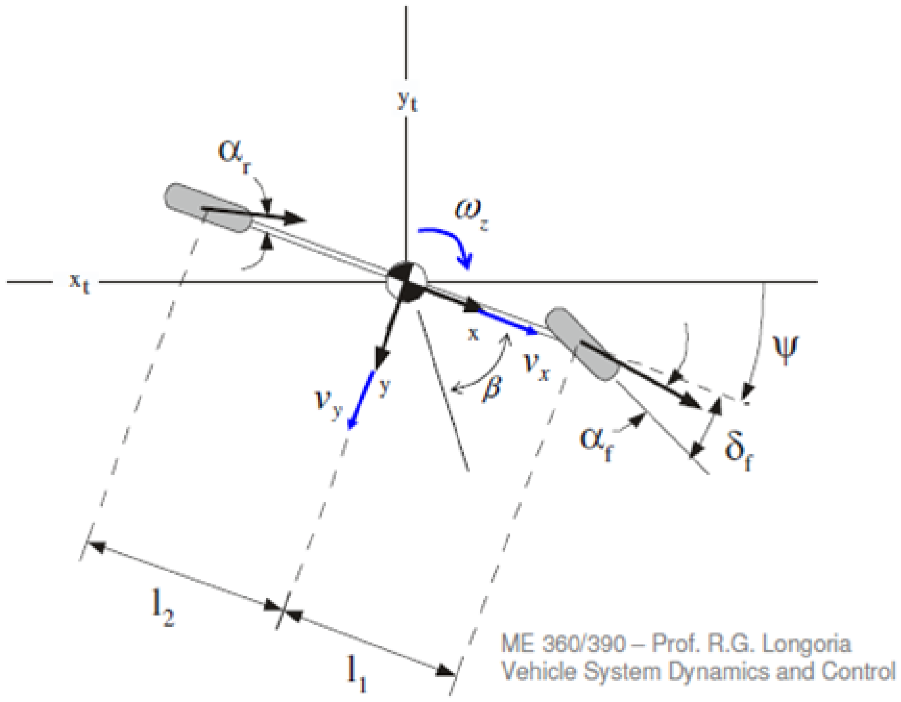


Figure 2.1: The three degree of freedom Bicycle model is a nonlinear dynamic model derived from physics and a simplified geometrical model [87].

The Bicycle model is a nonlinear single-track dynamic model which approximates a ground vehicle as a generic two-wheeled mechanical frame [38, 90]. This generates a series of nonlinear differential equations, given by equations 4.1-4.3. These equations relate the linear accelerations and angular acceleration to the vehicle parameters and states. When supplied with a sufficiently accurate tire model, it is relatively predictive in most dynamic regimes [83]. Within the context of vehicle control, the bicycle model generally performs well. However, there are two intractable issues that prohibit the use of the bicycle model for the purpose of performing sensor pose estimation. The first is that the bicycle model, and any other physically derived model, is defined for the center of gravity of the vehicle. This is not a problem if the aim is to control the vehicle, but when attempting to emulate the output of a reference inertial measurement unit, it is highly unlikely that it is located at the center of gravity. Hence the reference IMU will output linear acceleration values that are different from those at the CG, and therefore a model of accelerations for the CG will not faithfully reproduce the IMU's measurements. The second issue is that, to use the acceleration model for extrinsic calibration, high dynamics are required to achieve the desired level of observability. However, it is precisely in regions of high dynamics that the bicycle model breaks down. This can happen for various reasons, including slip between the tires and the pavement, and unforeseen non-linearity in the tire reaction force. In any case, the bicycle model is not accurate enough under the dynamic regimes that are pertinent to the model identification task. While physical models present the advantage of incorporating information from known physical laws, the task of developing a sufficiently accurate model under high dynamics is daunting enough that alternative methods have been sought that promise to avoid the need for a fine-grained understanding of physics.

Techniques to construct a vehicle model directly from observed data fall under the umbrella of system identification. System identification is a discipline that is primarily concerned with the algorithmic determination of the structure and parameters of the dynamics of an unknown physical system [7, 104]. Methods of system identification can reasonably be divided into two categories: structured, and unstructured. Structured methods attempt to identify the mathematical form of the system equations before searching for the parameters that best fit the measured data. The degree of structure in any specific system identification procedure varies, but all structured methods have in common an initial model structure identification step that is performed before the parameters of the model can be estimated. Model structure identification attempts to find the mathematical form that is optimal or most likely to have produced the observed data [64, 110, 120]. This involves searching the space of real-valued functions, which is significantly more difficult than searching a vector space of parameters. While parameter spaces are finite-dimensional, the space of functional forms is infinite. Hence, the search space must be limited in some way in order to reach a satisfactory model within finite time constraints.

This search can be done in multiple ways. A common approach is to start with a good approximation to the truth, whether obtained from a physical model of the system, or by another data-based mechanistic model, and recursively improve the model structure by varying it some amount at each iteration of the procedure [112, 116, 121]. Another approach is to limit the space of potential functional forms to a small subset of candidate forms [114]. This set of candidate forms could be the polynomials of degree less than n, for example, or the set of trigonometric functions. The choice of the candidate set is usually performed by a human expert with some experience that will hopefully allow them to decide what form the system takes.

2.1 Recursive Prediction Error Method

The dynamic structure of the system can also be approximated by constructing a graph that models state transitions, and recursively updating the graph given new information until it can no longer be improved [76, 122]. This method, shown in algorithm 1, is known as the Recursive Prediction Error (RPE) method. RPE has the advantage that it can accommodate a more complex mathematical structure, while still enabling the model to be recursively updated to better fit observed data.

Algorithm 1: Recursive Prediction Error Algorithm

```
Result: Parametric model with minimized mean-squared error.
while stopping criterion remains false do
    Update the parameter vector $\phi_t = \phi_{t-1} + L_t^* \hat{\varepsilon}_t$
    Update the Gain matrix $L_t^* = P_t^* \Psi_t (S_t^*)^{-1}$;
    Calculate the Covariance Innovation Matrix $S_t^* = \Lambda + \Psi_t^T P_t^* \Psi_t$;
    Update the Covariance Matrix $P_{t+1}^* = P_t^* - L_t^* S_t^* (L_t^*)^T$;
    set stopping criterion;
end
```

The RPE method begins with an a priori graph model corresponding to an existing, but unsatisfactory model structure [122]. The inadequacy of the model's structure is first demonstrated by estimating the prediction error of M_{priori} on the parameters when the model is compared to observed data [76]. The source of the model failure is then diagnosed by the system analyst. This can be done by inspecting the model graph one node at a time to identify the transition between correct and incorrect regions of the model structure, or more recently, by analyzing a set of system-wide parameters β as a distinct vector containing information about its validity [104]. The final step of the procedure is the rectification step. In the rectification process, the model graph is appropriately updated to address its demonstrated failure modes.

2.2 ARMA and ARIMA Models

The most basic form of statistical models is the family of auto-regressive models. This class of model only specifies the dynamics of the system in terms of a set of measurements taken consecutively in time [89]. An auto-regressive moving-average (ARMA) model, in particular, is any auto-regressive model in which the value of the current state of the modelled system is assumed to be a linear function of the previous p output states and the previous q error states [69].

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{2.1}$$

Here X_t is the output state at time t, ε_t is the error state at time t, and ϕ_i and θ_i are the model parameters. The time lag operator, L, is defined to be a function which yields the previous value of the state, X_{t-1}, when given the current state. The equation can be written equivalently in time-lag notation as shown below.

$$(1 - \sum_{i=1}^{p} \phi_i L^i) X_t = (1 + \sum_{i=1}^{q} \theta_i L^i) \varepsilon_t \tag{2.2}$$

The ARMA model is a statistically sound approximation of any stationary stochastic process. However, stationarity is a property that is almost never found in real dynamic systems. To address this issue, the ARMA model must be differenced to the appropriate degree. Differencing is a process that calculates the value of a new time-series at time t as the value of the existing time series minus the previous value, essentially producing an estimate of its derivative. This operation can be performed on the output states any number of times to obtain an estimate of the n^{th} derivative of the time series. Theoretically this procedure should drive the auto-correlation function to zero, and therefore result in a new, stationary process in terms of the n^{th} derivative of the output states [12]. Models that introduce the degree of differencing as a separate model specification parameter are known as auto-regressive integrated moving-average (ARIMA) models [104, 12]. A model *ARIMA(p, d, q)* with p-order auto-regression, q-order moving-average, and d degrees of differencing gives the equation below.

$$(1 - \sum_{i=1}^{p} \phi_i L^i)(1 - L)^d X_t = (1 + \sum_{i=1}^{q} \theta_i L^i) \varepsilon_t \tag{2.3}$$

If the differencing process succeeds in making the error state distributions stationary, then the ARIMA model will be able to optimally estimate the system output. In general, however, differencing only brings the error distribution closer to stationarity, while not entirely eliminating it. This could be the result either of some time-dependence in the system parameters, or because the data sequence does not contain the information necessary to properly estimate the system output. In the latter case, the Box-Jenkins procedure discussed next can be used. Figure 2.2 shows the auto-correlation and partial auto-correlation graphs for a sample non-stationary time series.

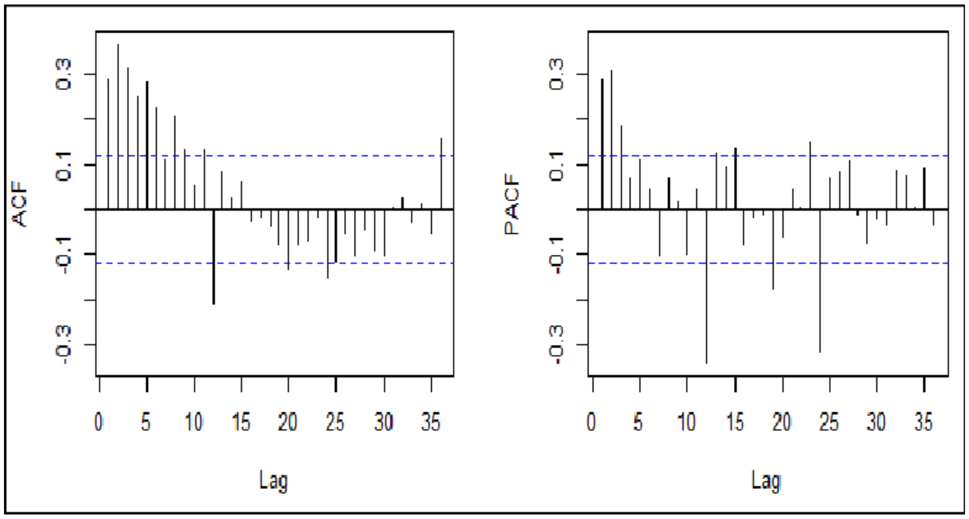


Figure 2.2: Sample auto-correlation and partial auto-correlation diagrams are non-zero for a non-stationary process [104].

2.3 Box-Jenkins Method

The Box-Jenkins procedure is an iterative three-step statistical estimation process to construct an ARMA or ARIMA model [12, 104]. First, as given in algorithm 2, a model structure is selected with special consideration taken to ensure the stationarity of the model given the measured data, and to account for any seasonality in the data. This is generally done by introducing a seasonality delay that corresponds to the period of the seasonal effect. By inspecting the spectral decomposition of the output states, the period m of the seasonal effect can be extracted. A parametric variable with delay m is then introduced to the model. The model is therefore able to take into account the value of the output state at time t - m if that measurement is the most strongly correlated with the current value of the state.

In Figure 2.3, for example, the auto-correlation function displays a strong seasonal effect. Inspecting the spectral decomposition of the time series reveals two strong peaks at around 0.01 and 0.09 hz. Hence, the Box-Jenkins procedure would recommend the addition of the state at t - 100 seconds and t - 11 seconds as predictor variables in the model.

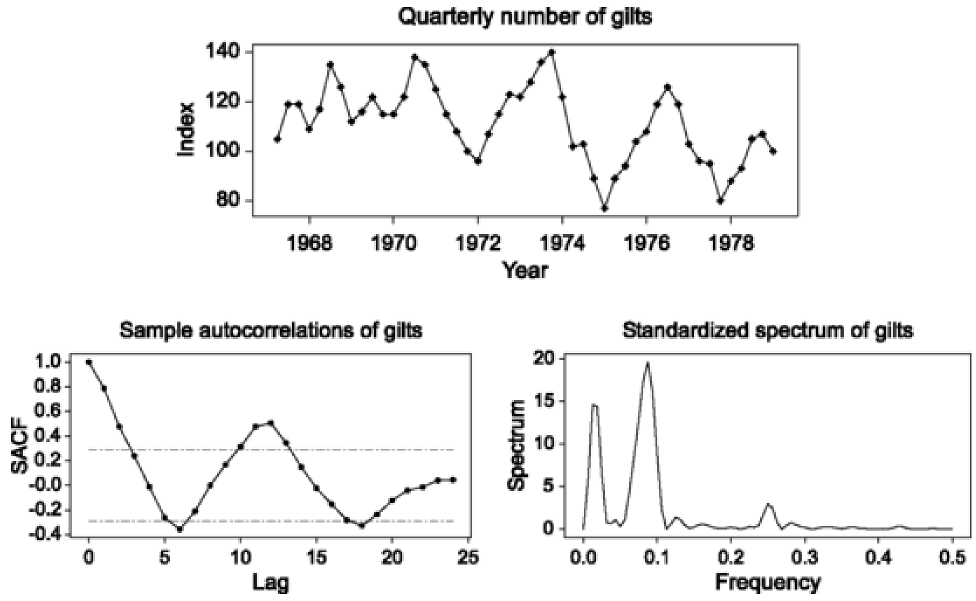


Figure 2.3: Partial auto-correlation and spectral decomposition for an example ARIMA model displaying strong seasonality [104].

Accounting for seasonal effects in the data will increase the stationarity of the error distribution. However, differencing can also be performed to drive the auto-correlation towards zero. Once the auto-correlation has been reduced as much as possible, the system parameters can be estimated using an appropriate criterion, such as maximum-likelihood estimation, the least-squares algorithm, a gradient-descent based optimizer, or any other parametric optimization tool [2]. In the third stage, the output estimates and the error states are recalculated, and the model is once more checked for stationarity and seasonal effects. The first two steps can then be repeated as desired until the model converges to a satisfactory level of estimation error.

```
Algorithm 2: Box-Jenkins Method
  Result: ARIMA Model with optimal time lag
  Apply the standard IV method to obtain an initial parameter estimate $\hat{\rho}$
  while stopping criterion remains false do
      Generate the instrumental variable time series from the auxiliary model
      $\hat{x}_t = \frac{B(z^{-1},\hat{\rho})}{A(z^{-1},\hat{\rho})} u_{t-\delta}$
      Obtain a new estimate of the noise parameter vector
      $\hat{\eta} = [\sum_{k=1}^{N} \hat{\psi}_t \hat{\psi}_t^T]^{-1} \sum_{k=1}^{N} (\hat{\psi}_t \hat{\psi}_t^T \hat{\eta} + \epsilon_t);$
      Prefilter the input, output, and instrumental variable time series;
      Compute the parameter estimate $\hat{\rho} = [\sum_{k=1}^{N} \hat{\phi}(t_k) \hat{\phi}^T(t_k)]^{-1} \sum_{k=1}^{N} \hat{\phi}(t_k) y_{f1}(t_k);$
      set stopping criterion = ;
  end
  Compute the parameter error covariance matrices $\hat{P}_{\rho} = \hat{\sigma^2}[\sum_{k=1}^{N} \hat{\phi(t_k)} \hat{\phi(t_k)}^T]^{-1}$
```

ARIMA models, especially when constructed with the Box-Jenkins procedure, can be a powerful tool for time-series estimation and forecasting. They are particularly robust to sudden disturbances to the system that cannot be accounted for by time-invariant physical models [104]. However, there are notable issues that have limited their estimation accuracy in real-world test cases, and that have motivated the development of alternative system identification methods. The first issue of note is that ARIMA models do not take into account the effect of exogenous inputs on the output states. This can, however, be remedied easily by adding an additional parameter vector corresponding to the known exogenous variables. A model of this variety is known as an Auto-regressive with Exogenous Variables (ARX) model. A second issue is that stationarity is never practically achievable in any real dynamic system. Hence, while ARIMA models promise statistical optimality, they can only approach this ideal. The accuracy of the model therefore depends as much on the skill of the model developer as it does on the nature of the dynamic system [104, 114]. Finally, the differencing process, while ideally reducing the auto-correlation of the output states, introduces significant error when there is noise or bias in the measurements. Since random noise is a property of any real data source, differencing tends to corrupt models more often than it benefits them.

2.4 Instrumental Variable Method

The Instrumental Variable (IV) method was introduced to address the fundamental issues affecting the performance of ARIMA models [104, 105, 111, 119]. In the Instrumental Variable method, an outside auxiliary variable, $z(t)$, is sought, called an instrumental variable, for which the explanatory variable is the only variable within the system that is affected by changes in the instrumental variable. In other words, the system variable X is only affected by the instrument through the error-source Y. Thus the instrument becomes a way of analyzing the the causal effect of X on Y. The choice of the instrumental variable vector depends both on the number of elements inside it, as well as how those variables are calculated from the measured data. In general, a set of IVs must be independent of all exogenous factors that also affect the output when the input is held constant. The choice of instrumental variable is, however, not immediately obvious. If it is known that there exists a measurable exogenous variable that only effects the output through a system variable, then this can be added to $z(t)$. However, such a variable is often not available. Instead, the IV vector is usually chosen to be some combination of filtered system inputs at previous times, with the consistency of the parameter estimate being the highest consideration. A parameter estimate is considered consistent if it approaches the true value of the parameter given enough data. This condition is stated by equation 4.4.

$$\hat{\rho} \rightarrow \rho_0 \; as \; N \rightarrow \infty \tag{2.4}$$

Instrumental variable methods are a generalization of the least squares algorithm for which the parameter estimate is modified to be consistent for an arbitrary disturbance [96, 101]. However, IV methods allow for consistent estimation, even when the explanatory variables are correlated with the error terms in the model [34]. This is accomplished by modifying the estimate so that it remains consistent for an arbitrary disturbance to the system. An ARX model is modified to contain the instrumental variable vector $z(t)$:

$$[\frac{1}{N}\sum_{t=1}^{N} z(t)\varphi^T(t)]\hat{\rho} = [\frac{1}{N}\sum_{t=1}^{N} z(t)y(t)] \tag{2.5}$$

where $\varphi(t)$ is the regressor, or explanatory variable vector, including exogenous inputs, $y(t)$ is the output variable vector, and $\hat{\rho}$ is the parameter estimate vector. When $z(t) = \varphi(t)$, this equation reduces to a least squares estimation.

The Instrumental Variable method can be expanded into the multi-step algorithm show in algorithm 3, which is, in practice, as accurate as an ideal instrumental variable method, i.e. one with a perfect choice of the auxiliary variable vector. The method can also be extended by making the estimation process iterative-recursive. The resulting system identification method is known as Refined Instrumental Variable (RIV) [104, 119]. Introduced by Peter Young in the 1980's, RIV is able to identify parameters in both discrete-time and continuous-time transfer function models, and is asymptotically equivalent to statistically efficient estimators such as maximum likelihood estimation [117, 118]. First, a simple IV model is constructed, and the system parameters ρ are estimated in the standard way. The output states are then estimated using the model and ρ, and the error between the measured output states and the estimated values is calculated. A prefilter is applied once more to the input states $u(t_k)$, output states $y(t_k)$, and output estimates $\hat{x}(t_k)$ as shown below.

$$f_1(z^{-1}, \hat{\rho}^{j-1}, \hat{\eta}^j) = \frac{C(z^{-1}, \hat{\eta}^j)}{D(z^{-1}, \hat{\eta}^j)A(z^{-1}, \hat{\eta}^j)} \tag{2.6}$$

Then ρ is updated once more to minimize the squared error in the estimation using the equation:

$$\hat{\rho} = [\sum_{k=1}^{N} \hat{\phi}(t_k)\hat{\phi}^T(t_k)]^{-1} \sum_{k=1}^{N} \hat{\phi}(t_k) y_{f1}(t_k) \tag{2.7}$$

where $\hat{\phi}(t_k)$ is the instrumental variable vector evaluated for a measurement at time $t = t_k$. The Recursive Instrumental Variable method can converge to the correct parametric solution within 3-4 iterations in most cases [117]. For this reason, it is a powerful method for optimally estimating the parameters of a known discrete or continuous-time transfer function model. However, its accuracy heavily depends on the user's choice of the prefilter, the instrumental variable vector, and the state covariance matrix. Without employing expert knowledge about the system in question, a sufficient level of accuracy for the resulting IMU emulation cannot be guaranteed. Despite its statistical optimality, the instrumental variable method lacks the degree of autonomy needed to solve the model identification problem raised in this thesis.

```
Algorithm 3: Refined Instrumental Variable Method
Result: Instrumental Variable Model with optimal convergence properties
Apply the standard IV method to obtain an initial parameter estimate $\hat{\rho}$
while stopping criterion remains false do
    Generate the instrumental variable time series from the auxiliary model
    $\hat{x}_t = \frac{B(z^{-1},\hat{\rho})}{A(z^{-1},\hat{\rho})} u_{t-\delta}$
    Obtain a new estimate of the noise parameter vector
    $\hat{\eta} = [\sum_{k=1}^{N} \hat{\psi}_t \hat{\psi}_t^T]^{-1} \sum_{k=1}^{N} (\hat{\psi}_t \hat{\psi}_t^T \hat{\eta} + \epsilon_t);$
    Prefilter the input, output, and instrumental variable time series;
    Compute the parameter estimate $\hat{\rho} = [\sum_{k=1}^{N} \hat{\phi}(t_k) \hat{\phi}^T(t_k)]^{-1} \sum_{k=1}^{N} \hat{\phi}(t_k) y_{f1}(t_k);$
    set stopping criterion = ;
end
Compute the parameter error covariance matrices $\hat{P}_\rho = \hat{\sigma^2}[\sum_{k=1}^{N} \phi(\hat{t_k}) \phi(\hat{t_k})^T]^{-1}$
```

2.5 State-Dependent Parameter Models

IV and Box-Jenkins models are useful for identifying non-stationary linear stochastic processes. However, they break down in the context of non-linear dynamics. In these regimes, linear approximations fail to model the system behavior, resulting in significant estimation error [104, 113]. State-Dependent Parameter (SDP) models extend the concept of Time-Variable Parameter (TVP) models to systems in which the parameters are not just dependent on time, but also on the state of the system itself. SDP, like TVP, employs recursive estimation and back-fitting to approximate the parameter as a function of the system's state vector. SDP deals with non-linearity, while extending the framework of ARIMA and ARX models, by considering a basic linear model, in which the parameters are modified to be dependent on the system state [116]. This model takes the form of equation 4.8.

$$f(\mathbf{X}) = f_0 + \sum_{i=1}^{N} f_i(X_i)X_i + \varepsilon \tag{2.8}$$

The choice of the parametric function f_i is at the discretion of the model developer. Generally it is chosen to be a low-order polynomial. This choice is considered appropriate because any smooth function can be approximated by a Taylor series around the desired point. A popular choice is the forced logistic equation shown below.

$$f_i(X_i) = \alpha X_{i,t} - \alpha X_{i,t}^2 + u_t \tag{2.9}$$

Hence SDP amounts to finding the correct coefficients of the Taylor series in the region of interest within the state space. The parameters are estimated iteratively by the following process: a Monte Carlo computer simulation of the system is run, generating a predicted system output that can be compared to the measured output. A fixed-interval smoothing filter is then applied, along with back-fitting, to adjust for disturbances to the system. Nonlinear optimization can then be used to search for the optimal parameter estimate. As in other classical system identification methods, the criterion for the optimization is either a maximum likelihood estimate, assuming univariate gaussian noise properties, or the mean squared error.

The basic form of SDP assumes that each parameter is dependent on only a single system state. However, it can easily be extended to models in which the parameters are a function of multiple different system states. The resulting models are known as Multi-State Dependent Parameter (MSDP) models. MSDP models generally take the form of a low-order multidimensional Taylor series approximation. MSDP is performed in the same way as SDP, except that at each point the trajectory through the state space, multiple estimates of the next value are generated and are pooled to provide an estimate of the mean and variance of the system state at that point in time [104].

SDP and MSDP enable the model developer to model systems containing a non-negligible state dependent non-linearity. However, because of the freedom they give to the developer in the choice of $f_i(X_i)$, they are also not very well suited to be employed autonomously. They can also suffer from collinearities of state variables, as well as singularities, if the model is not formulated properly. However, these problems are minimal in comparison to the problems that SDP have when dealing with significant non-linearities that vary swiftly in time. If the nonlinear form of the state dependency is not chosen correctly, then the parametric approximation generated by SDP will break down as soon as the system leaves the region of the state space in which the approximation was made [21]. Adding a higher order term to the Taylor series approximation may improve the model's accuracy in these regions. However, higher order polynomials are sensitive to small variations in their coefficient, and hence are not very amenable to any standard method of nonlinear optimization.

2.6 Comparison of Structured and Unstructured Methods

Structured methods have the major advantage of limiting the dimensionality of the parameter search space. However, the structure of the system must first be identified in order to use them, and this is a major obstacle to their autonomous deployment. Model structure identification is a complex task that involves a search over the infinite space of functional forms [104]. Selecting a small set of candidate structures is critical to the success of the identification process. However, the expertise of a human developer and knowledge of the global characteristics of the system is required to perform this step correctly. At present, neither can be feasibly automated. Perhaps the greatest advantage of structured models is that they can incorporate physical knowledge about the system, such as conservation of energy, conservation of momentum, or system geometry. This also leads to very compact and descriptive mathematical formulations, which can be understood and analyzed easily.

However, if there is an issue with the fundamental structure of the model, it can be very difficult to identify, and even more difficult to modify without starting the process over from the beginning.

Unstructured methods, in contrast, avoid the problem of structure identification entirely. In these methods it is assumed that the system's structure can be approximated using a model with very little structure but high symmetry and a large number of parameters. Hence there is no need for a separate structure identification step, or for an estimate of the covariances between the system variables. Unstructured methods also allow the model to be modified easily to adjust for error, whether by an iterative estimation process, or by a learning algorithm, which makes them flexible to deployment in different contexts. These properties make them more amenable to autonomous development than structured models. However, there are a number of issues that affect unstructured models. Unstructured models are also known as "black box" models, which means that they give little insight into the relationships between modelled variables, and their relative significance [104]. It is difficult to identify potential issues with the model, and hence it is hard to address systematic problems. Improving the model accuracy, beyond what it naturally converges to, is virtually impossible for this reason. In contrast to physics-derived models, they can also violate physical laws.

Black-box models also tend to exhibit poor generalization because of this non-physicality. Where structured models are more likely to be robust to unanticipated data sequences, they often break down radically outside of the region in which data has already been observed [61, 104]. In a similar vein, a large quantity of data is required to ensure the accuracy of the model within any particular region of the state space [95]. To make the model accurate for all regions of the state space may potentially require an infinite quantity of data. Black-box models, due to their large number of parameters, have the potential to overfit the observed data, resulting in systematic error when deployed on test data. Another issue is that the model architecture must also be properly selected for the task at hand. Although this is much easier than selecting the correct mathematical structure for a semi-empirical physical model, it still must be noted as a necessary step in the process. Despite these issues, unstructured models can be just as, if not more accurate, than structured models, and can be performed with a much higher degree of autonomy. This makes them a compelling choice for deployment on self-driving cars, and autonomous vehicles in general.

2.7 Automatic Selection

A compromise between the highly structured block diagram models and completely unstructured parametric models is Automatic Selection. Automatic Selection as a term refers to any system identification method which uses an algorithm and statistical metrics to choose the form of the nonlinear model from among a set of candidate models [17, 21, 31, 48, 104]. The most basic approach is to assume the model takes the form of a polynomial in k relevant variables selected from K candidate variables. This generic structure is often a good choice because polynomials are useful for approximating nonlinear functions, in particular with the Taylor series approximation. This is demonstrated by the Weierstrass approximation theorem, which states that any continuous function on a closed and bounded interval can be approximated by a polynomial to arbitrary accuracy [98]. However, other general structures can be chosen if they are likely to fit the observed data better. For example, a spectral decomposition might suggest the use of sinusoidal basis functions instead. Encompassing tests of the assumed nonlinear form can be performed to validate the choice of base class for the model if deemed necessary. For the general unrestricted model, the assumption is made that there is a local data-generating process that takes the abstract form shown below.

$$y_i = f(x_{1,i}, ..., x_{k,i}; \phi) + \varepsilon_i \tag{2.10}$$

Choosing the base class for the model to be the polynomials, the Taylor series representation of the model becomes:

$$y_i = \phi_0 + \sum_{j=1}^{K} \phi_{1,j} x_{j,i} + \sum_{j=1}^{K}\sum_{l=1}^{j} \phi_{2,j,l} x_{j,i} x_{l,i} + \sum_{j=1}^{K}\sum_{l=1}^{j}\sum_{m=1}^{l} \phi_{3,j,l,m} x_{j,i} x_{l,i} x_{m,i} + \sum_{j=1}^{N} \delta_j 1_{j=i} \quad (2.11)$$

where the representation has been truncated to the cubic term to prevent the number of parameters from growing to an unmanageably large number. This representation also includes a series of impulses, δ_j, that correspond to potential disturbances at each observation i, which enables outliers in the data to be detected and excluded from the model. Before the parameter space is searched, the potential non-linearity of the model is tested using a low-dimensional portmanteau test. The portmanteau test presents a hypothesis that the data is generated by a non-linear process, and compares this hypothesis to the null hypothesis that the process is linear. If the null is accepted, then nonlinear techniques are unnecessary, and a linear model is selected. If the null is rejected, then a significance test is run to prune unnecessary terms before nonlinear optimization can be applied to the parameters to arrive at a fully specified model.

Empirical studies demonstrate the effectiveness of Automatic Selection for nonlinear systems in a variety of disciplines [20]. However, the approach does have a number of noticeable problems. Automatically selected models can suffer from collinearity when the linear functions and nonlinear functions coincide. In these cases, multiple variables collapse into one, and the predictive power of the model is reduced. To address this problem, models are generally checked for collinearity and designed for the maximum possible orthogonality. Another problem is that 3^{rd} order or higher polynomial terms can cause rapid growth in model error. Outliers in the data can suggest the significance of nonlinear terms that in reality are not significant, and this can lead to drastic fragility in the model. While significance tests reduce the probability of this issue arising, they cannot completely eliminate the possibility. A last problem arises when the non-linearities in the dynamic system are very complex. Automatic selection is designed to approximate nonlinear functions that can be described by a relatively simple structure. If the function is highly nonlinear, then the approximation will be invalid in at least some regions of the system state space. However, the general form is fixed at the beginning of the identification process and cannot adapt to the degree of observed non-linearity. This issue motivates the development of models that can handle arbitrarily levels of non-linearity.

2.8 Neural Networks

Neural Networks are a class of unstructured nonlinear models that are specified using only observed data and a learning algorithm [43, 95, 97, 104]. They are well tailored for adaptive learning due to their uniform neural composition, where more structured mathematical formulations are less flexible. There are many varieties of neural network models. The Multi-layered Perceptron (MLP) is the most basic class of neural network model [14, 47, 82]. It consists of a set of hidden layers connected by a weighted linear function and a chosen non-linear function, as shown in Figure 2.4. In a forward pass, the input vector or regressor is fed through each of the hidden layers in series, to produce an output [13]. In the context of system identification, this output is a prediction of the system output state at the next epoch [23]. The parameters of the model are updated through a backward pass of the error through the model. In a backward pass, the error between the model prediction and the true value of the output state is calculated. Crucially, the neural network model is differentiable, which allows the gradient of the error to be calculated with respect to the model parameters, and this gradient is propagated backwards through the network [18, 41]. The model parameters are then updated using the gradient according to a learning rule.

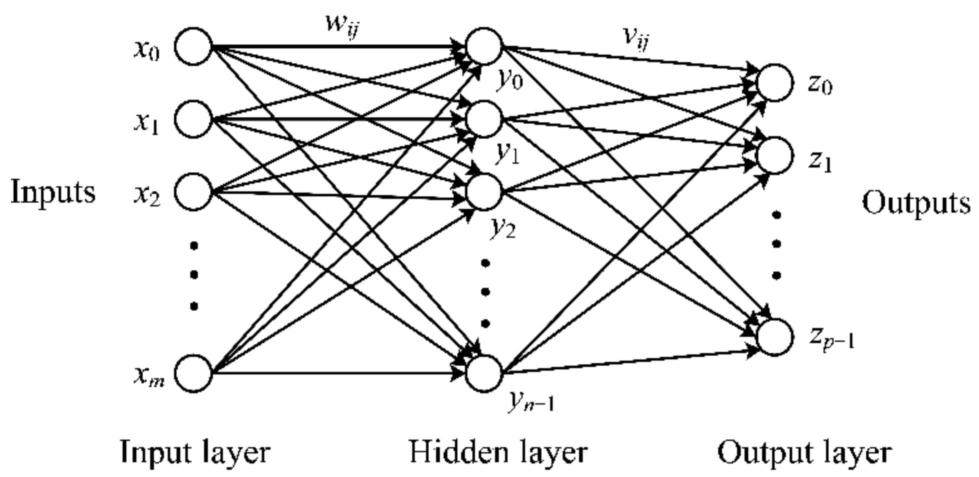


Figure 2.4: A standard neural network architecture [42].

2.8.1 Recurrent Neural Networks

Given a deep enough architecture, enough neurons per layer, and sufficient training data, an MLP can approximate a nonlinear function of any degree of complexity to arbitrary accuracy [9, 27]. This makes neural networks a very powerful tool for nonlinear approximation. However, in order for that approximation to be valid, the distribution of the output state on the regressor must be stationary. If the regressor does not contain enough information to estimate the output state, then the neural network will not be able to learn a mapping between the two. Recurrent Neural Networks (RNN) are designed to address this issue by extending the input state vector of the multi-layered perceptron to include previously estimated values of the target function [40, 109]. Figure 2.5 shows how the output states of hidden layers are fed back into previous layers via feedback connections.

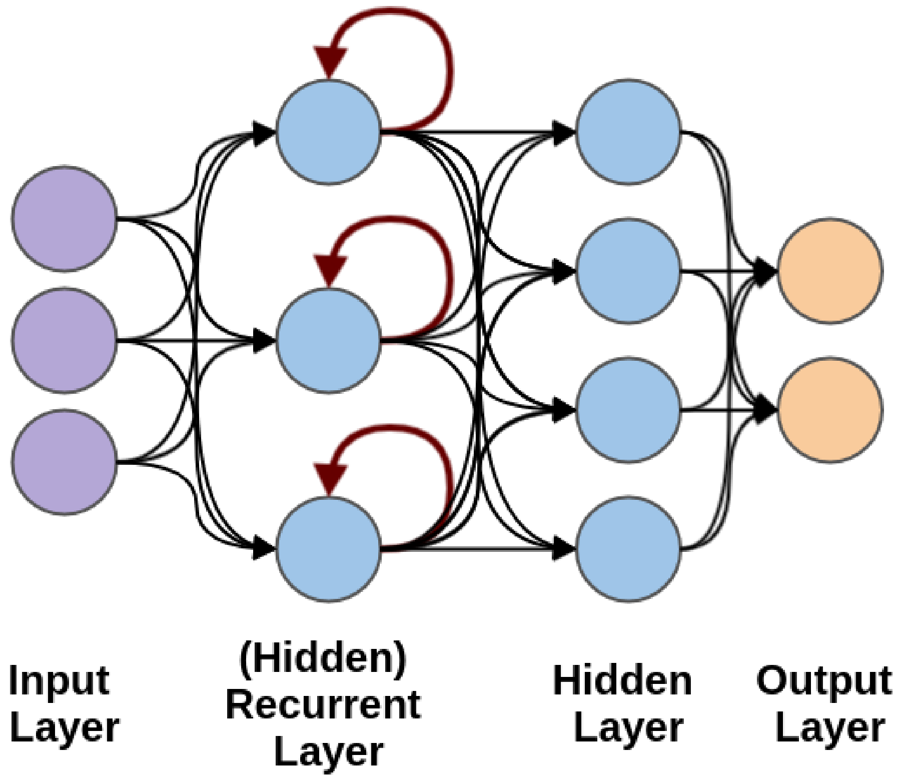


Figure 2.5: The Recurrent Neural Network architecture contains feedback connections between hidden units [30].

The current state is modeled as dependent on a history of the previous states of the system [32]. Hence the general unrestricted form of the RNN is the same as that of an ARX model. The RNN cannot be trained by standard back-propagation because of the feedback loop between its output and its input. Instead, it is trained by back-propagation through time [33]. The RNN is unfolded, giving the activations of each node at each point in time for which the network outputs a prediction. The gradient is then propagated normally across the unfolded network [40]. However, in order to reach long time lags, the gradient must travel across many unfolded layers. This can result in the "vanishing gradient problem"; the magnitude of the gradient is exponentially diminished at large time lags, and hence the update to the network has a negligible effect at this depth. If there is predictive phenomenon that occurs with some large time lag, the RNN will not be able to update its parameters to reflect that.

The Long-Term Short-Term Memory network (LSTM) is a time-series forecasting neural network model that incorporates the recurrent structure of the RNN and adds to it a "forgetting gate" that enables the model to pass gradients unchanged backwards through time [47, 50]. An LSTM is composed of cells which combine the current value of the input vector through an input gate with the previous value of the cell state through the forgetting gate. The activation of the hidden node is then calculated by transferring the cell state through an output gate. Figure 2.6 shows the internal behavior of an lstm cell. The leftmost sigma block is the forget gate. The forget gate weights the remembered value of the previous cell state c_{t-1} in the calculation of the new cell state c_t. The gate to its right in the diagram is the input gate. It weights the value of the cell input vector from the previous layer in the calculation of the new cell state. Finally, the output gate weights the value of the new cell state in the calculation of the input vector that is passed into the next layer of the network.

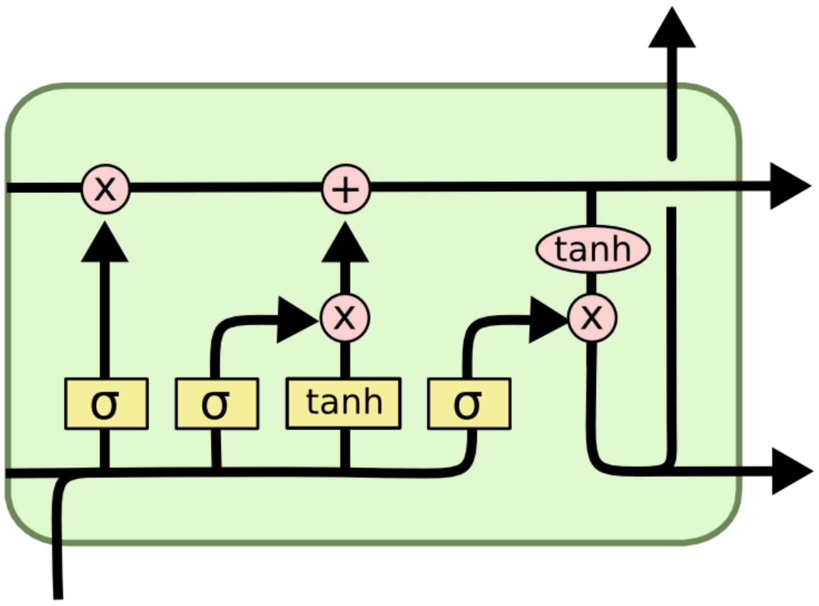


Figure 2.6: LSTM cell with input, output, and forgetting gates [66].

The cell-structure of the LSTM allows it to model both short term and long term dependencies. For this reason, it is the neural model of choice for many applications. However, because it is essentially equivalent to an MLP unfolded in time, it suffers from the same drawbacks. LSTMs require a large amount of training data in order to learn patterns in time series, due to their depth, and the extra parameters per neuron added by the forgetting gate and output gate. They also have a tendency to overfit the training data, leading to degraded performance at test time. This issue is exacerbated in the context of noisy and biased measurements, such as those from an IMU.

During training, the network may learn patterns that are the result of noise or bias rather than the actual system dynamics [41, 47]. Because the LSTM feeds forward through time, initial noise or bias can propagate through the whole estimate, causing increasing deterioration over time. This problem is magnified by the fact that the output estimate is fed back into the network. If the previous estimate was drastically incorrect, then the current estimate will be based on faulty information, and likely be even worse. For such regimes, a measurement model that accounts for noise and bias is necessary, or a nonlinear model that is robust to noise.

2.8.2 Basis Function Neural Networks

Neural Network models can be divided into two categories: deep layered networks, such as the MLP, RNN, or LSTM, and radial basis function neural networks. Radial basis function neural networks are a class of shallow neural networks that combine the concept of basis functions from automatic selection with the uniform structure of the multi-layered perceptron [47, 85, 104]. A set of radially-symmetric functions are placed in the input space and their parameters are varied until the error between their sum and the truth is minimized [84]. Radial Basis Function Neural Networks (RBFNN) have a long history of use in the realm of system identification for several reasons. Among neural network models RBFNNs are the most robust to noise. RBFNNs are also more explainable than deep neural networks, because they only contain a single layer. The effects of each basis function or neuron can thus be linearly separated, where the effects of each neuron in a deep neural network cannot be separated. RBFNNs also do not suffer from the vanishing gradient problem like recurrent neural networks, because they take every observed data point into account simultaneously during training. Finally, they are also sparser than neural networks, so they tend to generalize better and overfit less.

A number of different basis functions can be chosen for the network. However, Wavelets, Polynomials, and Gaussian distributions are the most common. The number of basis functions is generally determined when the network is initialized. During training, the parameters of the basis functions are tuned until the error between the output and the estimate is minimized. The parameters of each basis function depend on the specific network being used. The wavelet basis function has a center point, a maximum height, and standard deviations that determine how the wavelet propagates in the vertical and horizontal directions. The mathematical form of the wavelet is shown in the equation below:

$$g(\mathbf{x};\mathbf{w}) = w_{\lambda+1} + \sum_{j=1}^{\lambda} w_j \Phi_j(\mathbf{x}) + \sum_{i=1}^{m} w_i x_i \tag{2.12}$$

Where $\Phi_j(\mathbf{x})$ represents the jth wavelet in the network as a function of the input states. The wavelets are superimposed on a linear model, so that they will only need to represent the deviations of the system from non-linearity. Each wavelet is computed as the product of a set of mother wavelets that correspond to a given input variable z_i, as shown below.

$$\Phi_j(\mathbf{x}) = \prod_{i=1}^{m} \phi(z_{ij}) \tag{2.13}$$

Each mother wavelet $\phi(z_{ij})$ has a dilation parameter w_{ij}^{ζ} and a shift parameter w_{ij}^{ε} that control how the wavelet is distributed. Hence the input to the wavelet can be written in the following form.

$$z_{ij} = \frac{x - w_{ij}^{\varepsilon}}{w_{ij}^{\zeta}} \tag{2.14}$$

Together, the shift parameters control the center location of a particular wavelet, and the dilation parameters control the period of the waveform in one direction relative to another. Along with the linear model weights, bias and the weights on each wavelet, the wavelet neural network has five independent sets of trainable parameters. These parameter sets can be seen in the architecture of the Wavelet Neural Network shown in Figure 2.7.

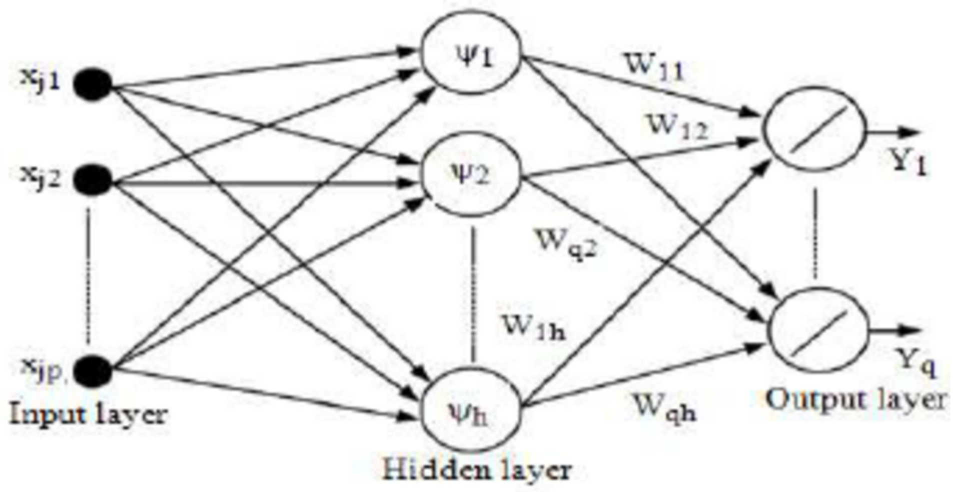


Figure 2.7: Wavelet neural network architecture [44]

The Pi-Sigma neural network is a basis function network with polynomial bases, which most closely resembles the automatic selection method of nonlinear modelling [3, 4, 10, 93]. The network is composed of alternating summation and product layers, followed by a sigmoid nonlinear activation function. This regular structure enables the Pi-Sigma network to efficiently model polynomial relationships between the input and output states. Figure 2.8 shows how the summation and product layers are arranged sequentially in the network architecture. The output is consequently a nonlinear function of the product of sums of the weighted inputs, plus an optional set of bias terms. Equation 2.8.2 relates the input to a layer of the Pi-Sigma network with its output.

$$y_i = \sigma(\prod_{i=1}^{n}\sum_{j=1}^{h} w_{jk}x_k + b_k) \tag{2.15}$$

The bias b_k can be applied at either the summation layer or the product layer. Since an nth order polynomial can be represented by the multiplication of n^{th} linear combinations of the inputs, the structure of the network results in a set of polynomial bases $w_{jk}w_{gh}x_kx_l$ that can approximate an arbitrary function in the same way that a Taylor series polynomial can. The weight sharing between these terms results in fewer overall network parameters, and consequently, faster training and a lower chance of over-fitting the training data set.

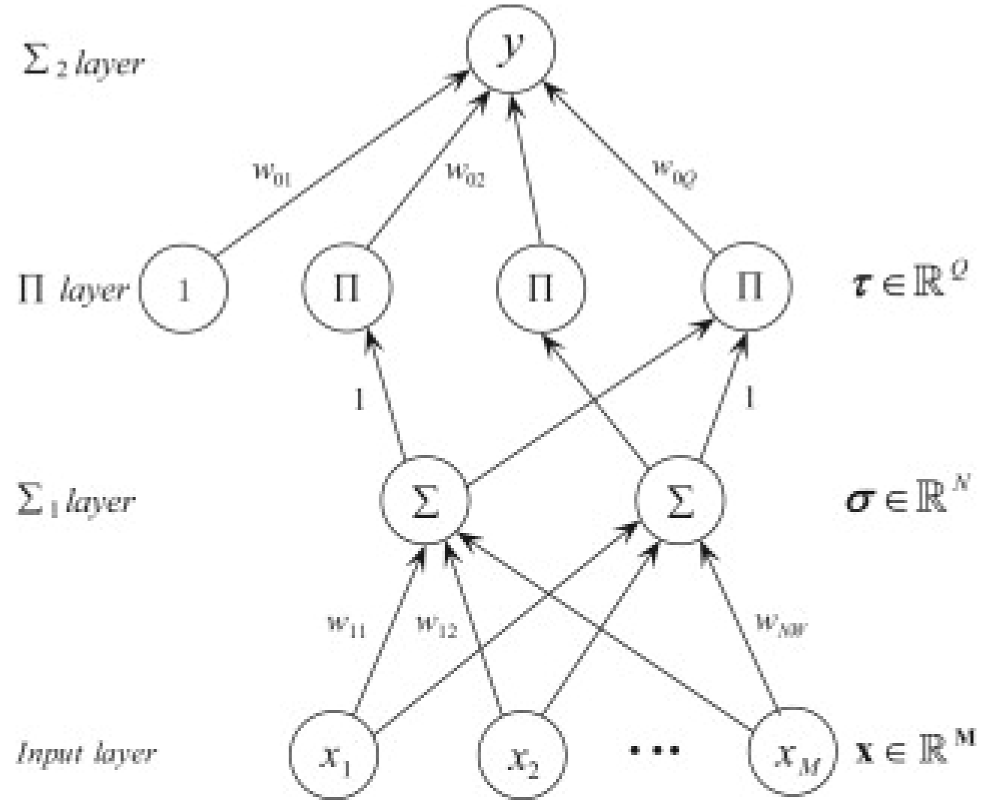


Figure 2.8: Pi Sigma neural network architecture [**pisigmann**]

For this reason, Pi-Sigma networks and other higher-order neural networks have received attention as an alternative to the MLP for modelling time series. However, the Pi-Sigma network does have some limitations. Weight sharing, while it does make training more efficient, also degrades the potential performance of the polynomial approximation, since a change in a weighted term will affect a number of other polynomial terms. Error grows rapidly in higher order terms, and this leads to unstable learning dynamics. Combined with the interconnected nature of the polynomial terms, the Pi-Sigma network has difficulty converging to a solution.

The Gaussian Radial Basis Function Neural Network (GRBFNN) is a neural network in which the form of the basis function is a radially symmetric gaussian kernel with mean μ_i, standard deviation σ_i, and height w_i [47, 85, 104], as shown below.

$$\phi(x_i) = w_i e^{\frac{-(x_i-\mu_i)^2}{\sigma_i^2}} \tag{2.16}$$

The gaussian form of the basis function is designed to facilitate local approximation of the nonlinear model. The value is maximized at the center location of the RBF and decreases radially according to a normally distributed random variable in order to capture the statistically noisy nature of real quantitative processes. MLPs, on the other hand, construct global approximations of the target function. Hence, RBFNNs tend to be more accurate when the process is confined to a small region of the state space. The shape of a gaussian radial basis function in 3D is shown in Figure 2.9.

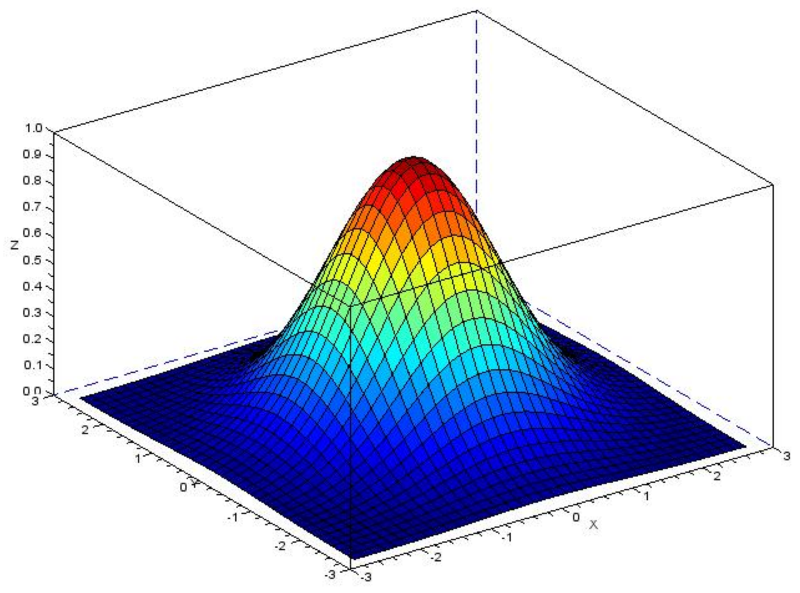


Figure 2.9: Gaussian Radial Basis Function in two dimensions [70].

Several factors contribute to make GRBFNNs robust to process noise. While their locally-concentrated nature makes them unable to generalize to previously unseen regions of the input space, it also prevents them from overfitting random noise in the input measurements. GRBFNNs also tend to be sparser than MLPs. Hence they have fewer parameters which can potentially over fit the training data. Like MLPs and other basis function networks, the GRBFNN is guaranteed to converge to any continuous nonlinear input-output mapping function by the universal approximation theorem [27, 45, 81]. However, they suffer in particular from the curse of dimensionality. The intrinsic complexity of a class of approximating functions such as radial basis functions increases exponentially with the dimensionality of the input space [65]. In other words, the space of approximating functions attainable with either MLPs or RBFNNs becomes increasingly constrained as the input dimensionality is increased. While an increase in the dimensionality of the model may result in an increase in the linear separability of the input space, it is moderated by an increase in the computational complexity of approximating the correct function [63].

Choosing the appropriate method of system identification for a particular application depends on the nature of the physical system, the degree of noise and bias in the measurements, prior knowledge about the properties of the physical system, the degree of non-linearity, the required degree of autonomy, and generality of the identification process. Structured methods are useful when there is strong evidence that the system dynamics conforms to a known mathematical structure, or when the explainability of the model is of high priority. They are also preferred when the observed data is sparse, and generalization becomes the most important attribute. Unstructured methods, on the other hand, are preferable when the structure of the model is poorly known, when there is a large or unlimited source of observed data, or when autonomy is of the utmost importance.

Chapter 3

IMU Emulation

3.1 Models Considered

The first stage of the calibration procedure is to emulate the output of an IMU centered on and aligned with the vehicle frame. Since the accelerometer contained in the IMU measures the linear acceleration in the vehicle frame, this task is effectively equivalent to modelling the dynamics of the vehicle at the chosen location. As discussed in the previous chapter, numerous methods exist to model system dynamics from observed data. Making an appropriate choice of the system identification method is crucial, because it will impact the performance of the resultant model, and hence the accuracy of the overall calibration, which depends on it. The best choice of method will depend on the characteristics of the system being modelled.

Physics based models have the advantage of incorporating information from physical laws, and are explainable in a way that unstructured models are not. However, they require exact identification of the model structure. While this may be easy for simple spring-damper-type dynamic systems, it is more difficult for complex systems such as ground vehicles. Vehicle modelling is a well-developed field, however. Complex models that take into account the vehicle suspension system, the slip between the tires and the road surface, and the lateral tire force generated by this slip, have been shown to provide accurate models of vehicle acceleration when its parameters are tuned for a specific vehicle.

However, the issue that must be confronted in this particular case is that these models are defined for the CG of the vehicle. The vehicle frame, on the other hand is centered on a chosen point along the longitudinal axis of the vehicle, generally at the center of the rear axle. The location of the CG relative to the origin of the vehicle frame is not known a priori.

Hence, the lever arm to the CG would need to additionally be estimated before the model could be used to estimate the lever arm and misalignment of the IMU relative to the vehicle frame. This introduces an extra variable to be estimated in the calibration procedure, which only serves as an additional source of error.

A State-dependent Parameter (SDP) model is another alternative that was considered for the IMU emulation. SDP models are more flexible than physics based models, while retaining model components that conform to the expected basic mathematical structure of the underlying physics. This method could also implicitly model the dynamics of the vehicle in the vehicle frame, rather than at the CG, if it contains parametrized components that correspond to the rigid body transformation of the accelerations. These properties make SDP an appealing candidate for the emulation stage. However, because this kind of model is linearized about observed points, it tends to break down in regions of high dynamics, where non-linear effects become important. The lever arm of the IMU is only weakly observable, because the rigid-body transformation effect on the linear accelerations is small. Hence, high dynamics are required to observe and properly estimate the lever arm. If an SDP model is used, it will only be employed in dynamic regimes where it is least effective.

A model that is inherently non-linear would not suffer from the same problem. This recommends the exploration of neural networks as a potential solution to the IMU emulation [97]. Neural networks are black box models, and their behavior is therefore difficult to explain, and errors in prediction are even more difficult to correct. However, they are capable of approximating any smooth real-valued function given sufficient training data and an architecture that contains enough learnable parameters. If a neural network were trained on the high dynamic regimes that will be needed during the calibration procedure, then it would learn to model the non-linear dynamics directly. Hence, the errors due to imprecise model structure that physics based models suffer from would be avoided, as well as errors due to linearization that hinder SDP. A wide variety of neural network architectures exist to perform the desired time-series prediction task. Among these options, two stand out: recurrent neural networks, and basis function neural networks.

RNNs are powerful tools for time-series approximation. ARX based RNNs have been used to forecast time series with high degrees of non-linearity that have proven impossible to model in any other way, such as chaotic laser emission data, and solar sunspot count [18, 86]. These results are a promising sign that an RNN could handle the precise IMU modelling in high dynamic regimes that are required for the calibration. This variant of neural network also takes into account past values of the target variable when making the prediction at the current time [40]. While this is useful in cases where the current state depends on the previous states, this property is not expected for the linear accelerations that are required to emulate an IMU. Previous values of acceleration should not affect the current value of acceleration; instead it depends solely on exogenous variables. Another effect of this property is that prediction error is sometimes amplified with time, as error in previous estimates of the target variable are compounded and accumulate. While this effect is negligible when the forecasting a single time step into the future, it becomes inevitable as the network extrapolates forward in time. For the calibration problem, direct measurements of previous target values are not available. Hence, the model must be able to predict many steps into the future. In practice this means that the extrapolation error effect becomes non-negligible, and the search for the optimal IMU emulation technique must continue.

Basis function neural networks are another class of system identification technique that were considered. Like standard neural networks, they are capable of approximating any non-linear function [27, 45]. What separates basis function neural networks from other classes of neural network is that they are shallow, and are composed of localized basis functions. The primary advantage of basis function neural networks is that they are robust to noise and bias. Since they have far fewer learnable parameters than conventional neural networks, they tend to avoid over-fitting the training dataset [11]. The localized nature of each basis function means that these networks learn local patterns more effectively. However, it also means that the network only learns the target function in the regions of the input space that were observed during training [108]. Due to this, a large amount of training data is required to train them. Despite this inconvenience, basis function networks have the most benefits of any of the available techniques, while not suffering from any serious drawbacks. For this reason, a gaussian radial basis function neural network was chosen for implementation of the IMU emulation algorithm.

3.2 Gaussian Radial Basis Function Neural Network

The IMU Emulation is performed by a Gaussian Radial Basis Function Neural Network. During training, the shallow network is constructed iteratively by placing radially-symmetric gaussian kernels in the input space until the training error is minimized [47, 86, 92]. Each kernel has a set of learnable parameters: a weight that determines the peak value of the kernel, a bias that increases or decreases the base kernel value, and a width that determines how concentrated the kernel is in the input space [79]. The set of kernels approximates the relationship between the input variables and the output variables, as shown in Figure 3.1. During testing, this relationship is recalled to estimate the output variables from the measured input variables [103]. In this implementation, the raw input variables are the steering wheel angle measured by an encoder, as well as the translational velocities and angular velocities obtained from a dual-antenna GPS. The GRBFNN outputs predicted linear accelerations in the x, y, and z directions for the vehicle frame centered at the chosen control point.

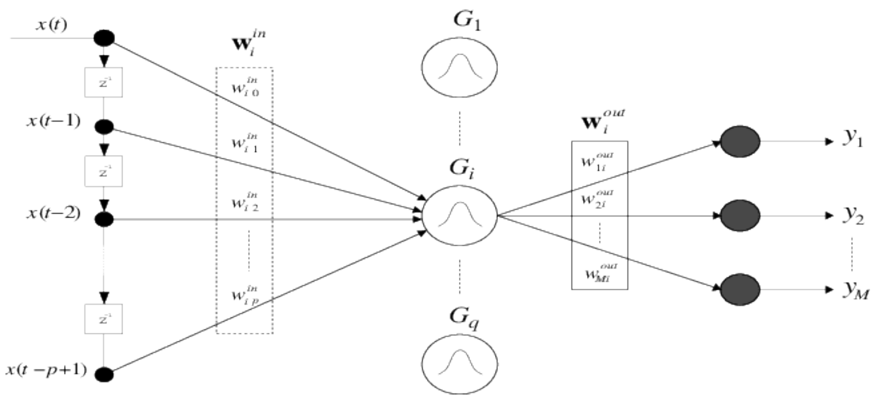


Figure 3.1: Gaussian Radial Basis Function Neural Network architecture [8].

The Gaussian Radial Basis Function Neural Network implemented in this thesis iteratively places a Gaussian kernel at a location in the input space, and optimizes the parameters of the kernel to minimize the resulting residual. In general the input space is a history corresponding to the previous n values of each of the input variables.

$$Hist = \begin{bmatrix} u_{1,t-n} & u_{1,t-n+1} & \dots & u_{1,t} \\ u_{2,t-n} & u_{2,t-n+1} & \dots & u_{2,t} \\ \dots & \dots & \dots & \dots \\ u_{k,t-n} & u_{k,t-n+1} & \dots & u_{k,t} \end{bmatrix}; \tag{3.1}$$

$$x = vec(Hist) \tag{3.2}$$

In this thesis, the input space was constrained to the current input state. This was done to limit the dimensionality of the input space, while retaining the variables that predict the target variable. Each kernel is placed at a set location in this input space, and takes the following form:

$$\varphi(x) = we^{-r\left\|x - \overline{x}\right\|} + b \tag{3.3}$$

Where $\varphi(x)$ is the kernel value at input vector x, w is the kernel weight, r is the width of the kernel, b is a constant bias, and $\overline{x}$ is the center location of the kernel in the input feature space.

The Gaussian kernel is a locally maximal basis function. For monotonic functions, i.e. those with no local maxima, the Gaussian basis functions generalize poorly. Another issue with Gaussian kernels is that their width is the same in all directions, which is sub-optimal if the function varies at different rates in different directions [11]. However, when the input space is normalized properly, the GRBFNN is able to converge to an accurate input-output model.

Algorithm 4: Training Algorithm for GRBFNN

```
Result: Neural Network that approximates the target function.
initialization;
while stopping criterion remains false do
    for 1:training loop size do
        initialize with parameters w_0, c_0, b_0;
        Optimize w, c, b, with Levenberg-Marquardt;
        Append w, c, b, to Network;
        Recalculate residual;
    end
    set stopping criterion = validate(network, validation dataset);
end
```

The Levenberg-Marquardt (LM) algorithm is used to optimize the parameters of each new basis function during training [86]. LM is faster than conventional gradient-descent optimizers, though it is less memory efficient. However, for the sparse gaussian radial basis function neural network, this is not an issue. LM behaves like a gradient-descent method when the current parameters are far from the optimal solution, which means that it converges slowly by taking a small step in the correct direction. However, when the current parameters are close to the optimal solution, it behaves like the Gauss-Newton method; the local error surface is assumed to be quadratic, and the location of the minimum is determined analytically. At each iteration of the training procedure shown in algorithm 4, the LM algorithm is used to search for the basis function parameters that minimize the residual between the target function and the output of the network.

Algorithm 5: Testing Algorithm for GRBFNN

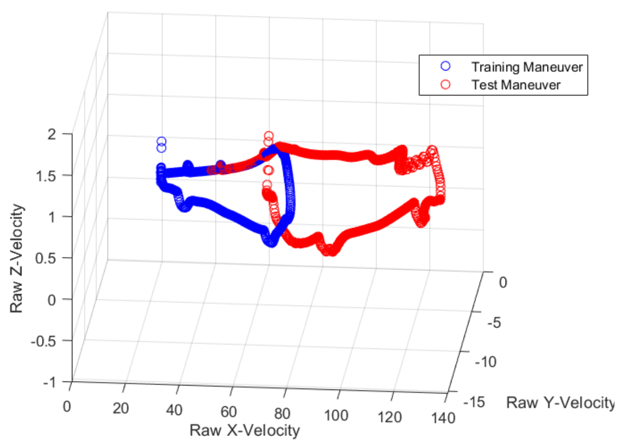
```
Result: Prediction time series for target dataset.
initialization;
for all points in test dataset do
    for all basis functions do
        prediction += basis function value at test point;
    end
    Prediction time series(t) = prediction;
end
```

At test time, the parameters of the network are fixed, and the relationship between the input space and the target function that was learned in training is recalled, as shown in algorithm 5. The predicted value for a test input vector $\vec{u}$ is the sum of the values of the individual basis functions at that location in the input space.

3.2.1 Input Feature Normalization

The Gaussian Radial Basis Function Neural Network is a universal approximator, in the sense that it can, given an infinite amount of time, estimate any differentiable function [80]. However, training on raw data rarely yields useful results. In poorly conditioned input spaces, training is inefficient, and can sometimes enter an asymptotic cycle, in which further training has no effect on the training error. Feature normalization attempts to ensure the efficiency of the algorithm, and prevent it from entering an asymptotic cycle [94].

Figure 3.2 demonstrates the effect of feature normalization on the input feature space of the GRBFNN. Training and test routines collected with different velocity profiles for a chosen test vehicle are shown as blue and red trajectories. Before feature normalization, the training and test routines occupy different regions of the feature space. However, the GRBFNN can only recall the target accelerations for regions of the feature space that have been spanned during training. It is therefore unable to recall the target function for the sample test routine. The normalization process maps the training and test routines to the same region of the input feature space. Hence, the GRBFNN is now able to recall the learned target function during testing after normalization.



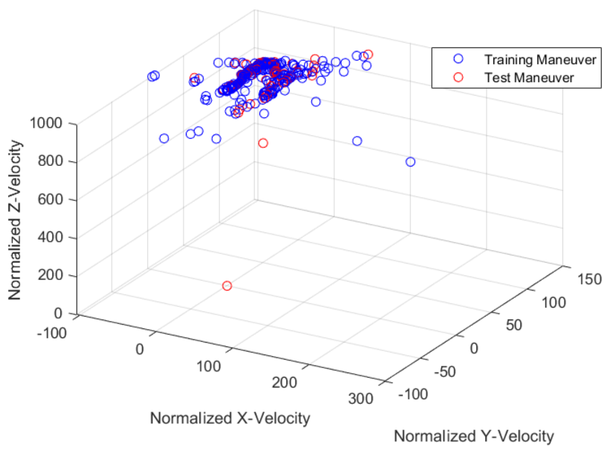


Figure 3.2: Raw (Left) and Normalized (Right) velocity feature maps for training and test maneuvers obtained with different forward velocities

The steering angle has a domain of [-180, 180] so this variable can be normalized to the range [-1, 1] using a simple scale factor. The first major problem encountered with the input space for the IMU emulation is that the velocity measurements can theoretically be infinite. Hence the input velocities were normalized to the magnitude of the velocity vector according to the following equations.

$$v_x^* = \frac{v_x}{\left\|\vec{v}\right\|} \tag{3.4}$$

$$\left\|\vec{v}\right\| = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{3.5}$$

Since vehicles tend to have a much greater longitudinal velocity than lateral or vertical velocity, the normalized velocity features map onto the same region of the feature space for training samples captured with drastically different forward speeds. The angular velocities measured by the inertial measurement unit can also be theoretically infinite. The cross-product of the angular velocity and linear velocity is taken, and then normalized to the magnitude of the same vector, to obtain a composite feature that is a better predictor of the ground truth linear accelerations. This feature is shown in the equations below.

$$\vec{a_c} = \vec{\omega} \times \vec{v} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} \omega_y v_z - \omega_z v_y \\ \omega_z v_x - \omega_x v_z \\ \omega_x v_y - \omega_y v_x \end{bmatrix} \tag{3.6}$$

$$a^*_{cx} = \frac{\omega_y v_z - \omega_z v_y}{\left\| \vec{\omega} \times \vec{v} \right\|} \tag{3.7}$$

The feature space is also stretched by a scale factor in order to smooth the optimization of the basis function parameters. This is a useful transform particularly when the observability of the target variable is low, which is often the case for the vertical component of the acceleration. A scale factor of 1000 for the input features was found to produce reasonable results.

3.2.2 Cross-Validation

A major issue with neural network models is that they tend to over fit the training data at the expense of performance at test time [6, 47, 86]. For basis function neural networks, over-fitting tends to begin after the number of basis functions exceeds the number of training data points. This problem is mitigated by the iterative training procedure used by the GRBFNN, as implemented in this thesis. Parameters are only added as needed, and hence the network generally remains sparse throughout training. However, the algorithm must decide to halt training at some point, to prevent over fitting of the training data at the expense of performance at test time. To make this decision, a validation procedure is generally employed, which checks the error on a reserved test dataset across training. If the validation error starts to increase while the training error is still decreasing, training is halted and the weights are fixed. Choosing the optimal time to stop training is a non-trivial problem, though, because the future behavior of the validation error is unknown a priori. An increase in the iteration-over-iteration validation error might indicate that over-fitting is beginning to occur, but it might also be a result of noise.

To address this problem, a cross-validation algorithm was implemented for the GRBFNN that checks if the validation error has reached an asymptote, or increased over a number of consecutive iterations. This was accomplished by fitting a line to the previous N error measurements. The validation routine is run on a held-out training dataset after every k training iterations. Once a minimum or an asymptote has been detected, the algorithm throws a flag to stop training. The network parameters for the optimal configuration, i.e. the one for which the recorded validation error was lowest, are taken and fixed before the testing stage.

Chapter 4

Pose Estimation

Once the IMU emulation has been performed, the gyro rates for the vehicle, and two sets of linear accelerations are available: one set measured by the IMU to be calibrated on the body of the vehicle, and a set of emulated accelerations for the control point at the center of the rear axle. Given this data, the pose of the physical IMU relative to the emulated IMU can be estimated using an IMU-to-IMU extrinsic calibration technique. Existing methods to perform IMU-to-IMU calibration include the Recursive Least Squares (RLS) filter, which adapts the estimated state at each iteration so as to minimize the mean-squared error between the input signals [54]. This method exhibits desirable properties, such as fast convergence. However, it assumes that the input signals are deterministic, while IMUs in the real-world display stochastic properties.

An alternative to RLS is the Least Mean Squares (LMS) algorithm, which applies a gradient descent optimization technique to search for the estimate that minimizes the mean squared error [115]. In contrast to the recursive algorithm, LMS can be applied to stochastic signals. However, it does not account for model error in the emulated IMU signal. If the emulation is drastically incorrect for a certain finite portion of the test routine, there is a risk of systematic error in the pose estimation.

4.1 Maximum Likelihood Formulation

To resolve this issue, a maximum likelihood search method was implemented. Optimization is performed over the space of possible pose estimates, using the likelihood of an estimate as its fitness metric. Unlike least squares, this metric allows information about the quality of the model's acceleration estimate to be incorporated when searching for the optimal pose estimate [49]. The likelihood of a given estimate $[\,\hat{r}, \hat{\theta}\,]$ of the IMU's pose is taken as the sum of the individual likelihoods of the accelerometer measurements in the test routine. This formulation is shown in equation 4.1

$$L(\,[\,\hat{r}, \hat{\theta}\,]\,) = \sum_{t=0}^{T} likelihood(\,\vec{a}_{s,t} \mid (\hat{a}_{cp,t}, [\,\hat{r}, \hat{\theta}\,])\,) \tag{4.1}$$

The likelihood of each measurement is a statistical measure of the probability that a stochastic acceleration signal with white Gaussian noise properties would produce the given measurement. The likelihood of any measurement given the modeled accelerations in the vehicle frame and the estimated lever arm and misalignment is:

$$likelihood(\,\vec{a}_{s,t} \mid (\hat{a}_{cp,t}, [\hat{r}, \hat{\theta}])\,) = log(2\pi) - \frac{1}{2\Delta u_t^2} \sum_{i=1}^{3} \Delta a_{i,t}^2 \tag{4.2}$$

$$\Delta\vec{a}_t = \vec{a}_{s,t} - \hat{a}_{s,t} \tag{4.3}$$

Where $\Delta\vec{a}_t$ is the difference between the acceleration vector measured by the IMU on the body of the vehicle $\vec{a}_{s,t}$, and the estimated acceleration, $\hat{a}_{s,t}$ for the same sensor, given the emulated acceleration vector for the control point, $\hat{a}_{cp,t}$, and the IMU pose estimate. The acceleration estimate is calculated by assuming a rigid-body transformation of the accelerations from the vehicle frame centered at the control point, to the sensor frame. The general form of this equation is:

$$\hat{a}_{s,t} = R(\hat{\theta})\,(\hat{a}_{cp,t} + [\,\alpha \times \hat{r} + \omega \times \omega \times \hat{r}\,]) \tag{4.4}$$

Where $R(\hat{\theta})$ is the rotation matrix formed using the estimated Euler angles $\hat{\theta}$. The likelihood of a measurement is weighted with a factor Δu_t that corresponds to the degree to which the IMU emulation is trusted to produce an accurate estimate of the true acceleration in the vehicle frame. It is calculated as the distance between the input state vector for a measurement in the test routine, and the nearest input state vector in the training dataset:

$$\Delta u_t = min[|(\vec{u}_t - U_{train})|] \tag{4.5}$$

Where U_{train} is the set of all input state vectors in the training data set. If the test vector is close to a training vector, then the GRBFNN can be expected to have learned the dynamics in the region and produce an accurate estimate. Conversely, if the test vector is far away from the nearest training vector, then the acceleration estimate can be considered to be unreliable. Consequently, this measurement is weighted less in the overall likelihood calculation.

4.2 Optimization

A covariance-matrix adaptation (CMA) based genetic algorithm is used to optimize the IMU pose estimate. CMA is a stochastic, gradient-free optimization technique that converges to a global maximum or minimum through a process of selection and mutation [58]. The algorithm is initialized with a small population of pose estimates. At each iteration, the population is updated according to a set of rules that prefers individual estimates with high fitness, and restricts the new population to a distribution that fits the covariance matrix of the population calculated in the previous iteration. The population update in CMA is illustrated in Figure 4.1. The orange dashed ellipse displays the covariance of the population distribution. As the population is updated, the covariance increases, allowing individuals to expand towards the global optimum. Once the global optimum has been discovered, the covariance decreases. This increases the density of the population within the region, allowing for a finer resolution search.

The covariance update to the population distribution ensures that the algorithm rapidly discovers local minima. By setting the initial variance of the population distribution relatively high, the algorithm will search the whole space for minima, greatly increasing the probability that it discovers the global minimum.

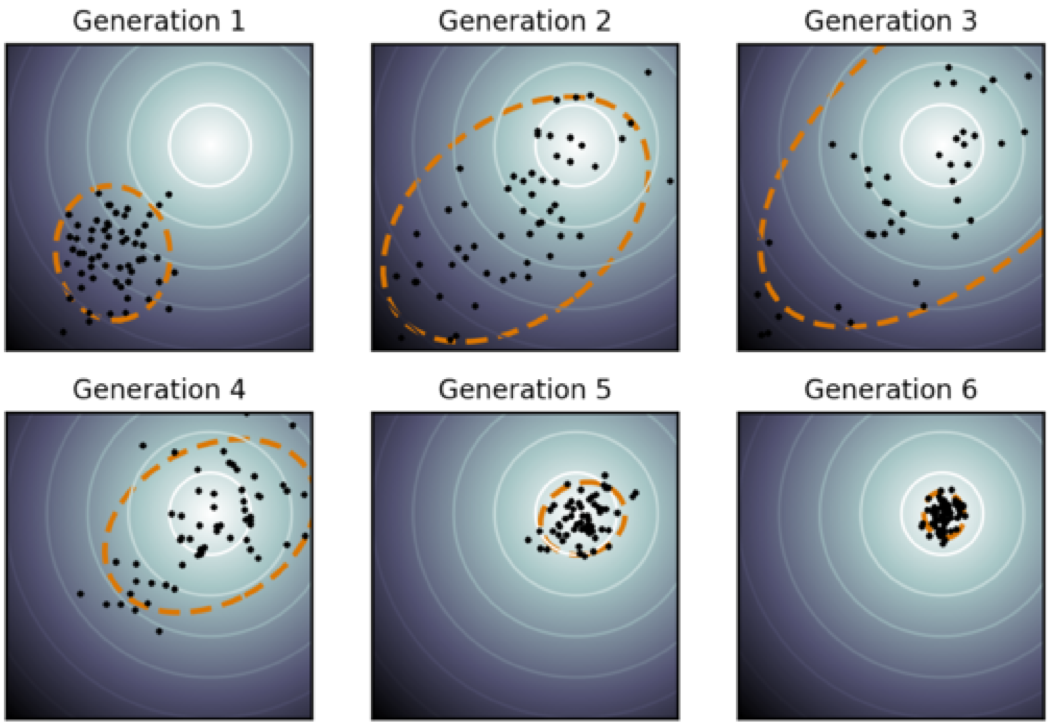


Figure 4.1: A covariance matrix adaptation evolutionary strategy adapts its population distribution at each iteration to conform to the error surface [24].

This method is critically important in cases where the input space is ill-conditioned, as arises when the dynamics of the vehicle are not fully observable during the test routine. Gradient-based optimizers perform poorly when the search space is ill-conditioned, because the gradient is much greater in magnitude in one direction than in all other directions [99].

This results in an unstable update of the estimate, which prevents the optimizers from converging to a solution. Gradient-based optimizers were tested and invariably failed to converge to the global optimum. CMA, however, always converged to the optimal solution in both the simulated and experimental trials performed in this thesis.

Chapter 5

Training and Test Procedures

The calibration procedure can be employed online and fully autonomously at any time, so long as a small number of requirements are met. Firstly, a minimal sensor array must be present on the vehicle. A steering angle encoder, a wheel odometry system, and two GPS antennas must be available and rigidly mounted on or in the vehicle in order to capture the raw input measurements that feed into the IMU emulator. Secondly, the calibration maneuver must contain a brief initial standstill phase as well as a straight section and a curved section. The standstill phase is required for the initial roll and pitch alignment. A curved section is required in order to achieve the desired observability to perform the full IMU pose estimation. Finally, the maneuver must be performed in clear sky conditions, so that the GPS antenna system provides consistently accurate velocities and angular rates.

No direct measurements of the extrinsic sensor parameters are necessary after rearranging the sensor suite. The only manual intervention required is to drive the calibration maneuver. The choice of this maneuver is relatively free as long as it meets the minimal requirements previously listed. However, before the calibration procedure can be employed on a vehicle, the IMU emulator must be trained to model that specific vehicle's dynamics.

5.1 Training Procedure

Algorithm 6 shows the procedure to train the GRBFNN to emulate the output of an IMU in the vehicle frame. Training the GRBFNN requires a ground truth for the input states, as well as the target accelerations in the vehicle frame. The first step of the training procedure is therefore to mount a high-grade IMU at the origin of the vehicle frame. Generally, the center of the rear axle, along the longitudinal axis of the vehicle, is chosen. This is the control point for many vehicles, to which the lever arm of the IMU will be estimated during the calibration procedure. The precise location and orientation of the ground truth unit can be measured with a laser surveying tool. The ground truth inertial measurement unit does not strictly need to be located at the control point, if the exact position of the unit relative to the control point is measured, as well as its orientation relative to a set of reference points on the vehicle. The combination of lever arm and misalignment angles can be used to transform the ground-truth IMU measurements into the vehicle frame, after the data has been collected. Hence, a perfect mounting of the instrument is not required.

The GPS antennas must also be mounted on the vehicle. Either a dual-antenna system, or two separate antennas, may be used. Once the antennas have been placed at arbitrary desired locations on the vehicle, the receiver positions are measured with the surveying tool. If a dual-antenna system is being used, the antenna lever arm should be known a priori. Otherwise, it should be calculated from the measured receiver position outputs.

Once it has been verified that the IMU, GPS antennas, steering wheel encoder, and wheel odometers are functioning, data can be collected in a series of training maneuvers. It is ideal if these maneuvers are performed in as few attempts as possible, because the bias of the ground-truth IMU will differ each time it is powered on [107]. These maneuvers should be designed to maximize the observability of the linear accelerations. Hence, high dynamic maneuvers such as figure-8s or 90 degree turns are recommended. Some vertical excitation, though less important, may also be required. The localized nature of the GRBFNN means that it will only learn the target function in regions of the input space that have been observed during training [47, 86]. It is therefore desired to span as much of the input space as possible across the training maneuvers. The normalization process reduces the effort involved in this aspect, because similar maneuvers do not need to be reproduced at multiple different velocities. However, varying the velocity and steering angle profiles should help explore the input space to a sufficient degree.

After the series of training maneuvers has been completed, the raw data from the steering wheel encoder, wheel odometers, ground-truth IMU, and GPS antennas can be extracted. Once the separate data sources have been synchronized in time, they can be concatenated into an observation vector time series. The steering angle and angular rates from the ground truth unit can be taken as is. The velocity from the GPS must be rotated into the vehicle coordinate frame using the reference survey measurements taken at the beginning of the training procedure. The vector can then be normalized as described, and the neural network can finally be trained on the normalized input vector. Three separate models are trained, one for each component of the linear acceleration that have been measured with the ground truth IMU.

```
Algorithm 6: Training Procedure
  Result: GRBFNN models that emulate linear accelerations in the vehicle frame
  (1) Mount the ground truth system at the control point, in-line with the
      longitudinal axis of the vehicle, or at an arbitrary location and orientation on the
      vehicle body.
  (2) If the ground truth system was mounted in an arbitrary pose, measure its
      location and orientation relative to the control point with the surveyor.
  (3) Mount two GPS antennas, or a dual-antenna system, onto the body of the
      vehicle. Measure the receiver positions with the surveyor.
  (4) If using two separate antennas, calculate the distance between the receivers.
  (5) Perform a series of maneuvers designed to maximize observability of the linear
      accelerations and span the normalized input feature space.
  (6) Form the input variable time series from the data collected during the training
      maneuvers. If the ground truth system was mounted in an arbitrary pose, the
      accelerations and angular rates need to be transformed into the vehicle frame.
  (7) Train a GRBFNN model to emulate each component of the measured linear
      acceleration in the vehicle frame.
```

5.2 Test Procedure

With a model trained to perform the IMU emulation in hand, the calibration procedure can be performed online and fully autonomously. Algorithm 7 details the steps necessary to employ the developed calibration technique. The IMU and two GPS antennas, along with any other sensors that are being used, must first be mounted on the vehicle at desired locations. After the sensors have been powered on, and all of the systems are running properly, the vehicle must wait in a standstill for a brief moment. 20-30 seconds was observed to be sufficient for this purpose. Then, the operator should drive a calibration maneuver that contains at least one curved section or turn.

```
Algorithm 7: Calibration Procedure
  Result: IMU-to-Vehicle relative pose
  (1) Mount the IMU and GPS antennas on the vehicle at desired locations.
  (2) Verify that all sensors are functioning properly.
  (3) Initiate the calibration procedure. Wait in a standstill phase for a short period
    of time.
  (4) Drive a maneuver that contains at least one straight section and at least one
    curved section.
```

The data captured during this maneuver will then be used to perform the IMU calibration online. If separate GPS antennas are being used, the relative position must first be fixed. From the straight sections of the maneuver, the misalignment between the vehicle and world frames is calculated. Then, the GPS velocities and angular rates are rotated into the vehicle frame using this alignment. With this information, in addition to the steering angle, the GRBFNN will emulate the linear accelerations in the vehicle frame for the calibration maneuver that has just been performed. The standstill phase is then used to give an initial estimate of the pitch and roll misalignment of the IMU. Finally, a covariance matrix adaptation based maximum likelihood search method estimates the full 6D pose of the IMU relative to the emulated IMU. This gives the lever arm and angular misalignments between the IMU and the vehicle frame that are the objective of the calibration procedure. The remaining sensors on the vehicle, whether they be LiDAR, radar, camera, or otherwise can then be calibrated relative to the IMU, and from there to the vehicle frame using the values that have obtained. Thus, with an IMU and a minimal array of sensors, the entire sensor suite of an autonomous vehicle can extrinsically calibrated online by driving a short maneuver.

Chapter 6
Simulation Results

The IMU emulation algorithm was trained on, and tested against, a battery of simulated datasets designed to probe the accuracy and generalization capability of the neural network model. These datasets were collected using the Carsim 9 engine and math model by choosing a combination of a steering and forward velocity profiles. Carsim is a high fidelity simulation engine that emulates vehicle dynamics with complex physics based semi-empirical models. In each test, the observability of the vehicle's dynamics was of paramount concern.

In the 'Rough' dataset, the training and testing data were generated with the same maneuvers, and at the same velocity, but with a differing degree of surface roughness. This test was designed to measure the robustness of the model to day-to-day changes in conditions and/or differences between the condition of the road surface on which the training data was acquired and the road surface on which the calibration routine is conducted.

The Step Steer dataset captures the vehicle dynamics for the same forward velocity, but with different steering profiles. The model is trained for a set of runs with constant steer angle of different magnitudes, and then tested against a run with a constant steer angle not observed during training. This test was designed to assess the network's ability to generalize across a number of high-curvature maneuvers that could potentially be observed during a calibration routine.

The ability of the neural network to handle noise in the input and target measurements was also tested. Noise rejection is a crucial aspect of any algorithm that is designed to use measurements from real-world sensors. Commercial grade IMUs in particular suffer from low signal-to-noise ratio. The noisy step steer dataset is identical to the step steer dataset in every other way.

The velocity difference dataset examines the model's robustness to differences in forward velocity between the test calibration maneuver, and the maneuvers observed during training. Hence it also tests the effectiveness of the magnitude feature normalization applied in the pre-processing step. The model is trained on a set of routines with the same steer profile, but with different forward velocity profiles, and is tested against a routine with a forward velocity profile that was not observed during training.

The final dataset investigated combines a constant steer angle difference with a constant forward velocity difference. This tests the network's ability to perform the IMU emulation task in the general case. Perfectly spanning the normalized input space during training is impossible. It is therefore important that the GRBFNN be able to generalize as much as possible to maneuvers that are different in both velocity profile and steer angle profile.

Input and Target Time Series

The simulated data sets were designed specifically to maximize the observability of all the available input variables. This demand resulted in high-dynamic maneuvers capable of exciting the vehicle's velocity in the vertical direction, which is generally the least observable variable for a vehicle operating under normal conditions. The high dynamics cause the simulation engine to generate peaks and oscillations in the simulated data, that may or may not occur in real-world tests. However, the wide-variability in the input variables allows a greater region of the input space to be spanned with one data set.

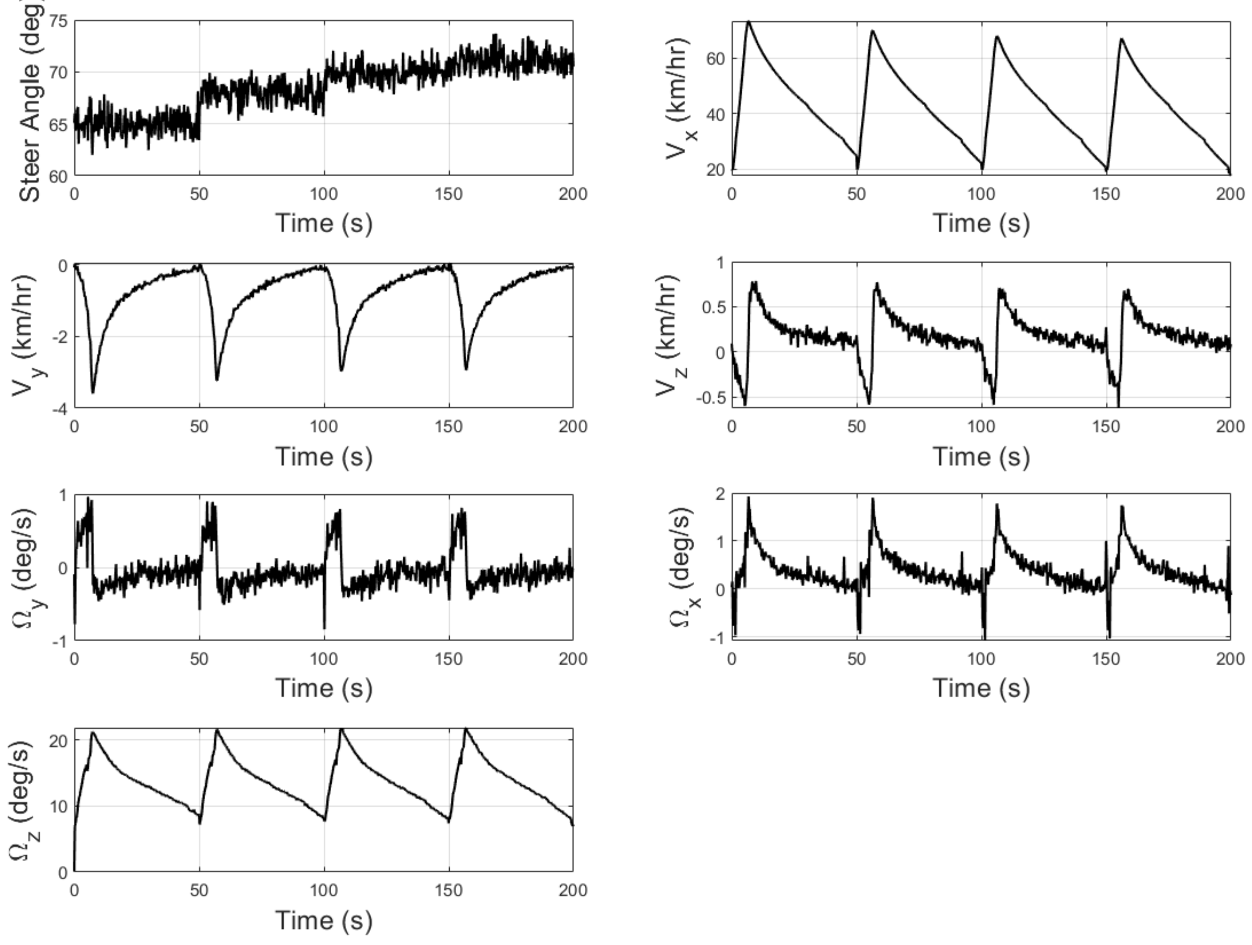


Figure 6.1: The raw input time series from the Noisy Step Steer Data set.

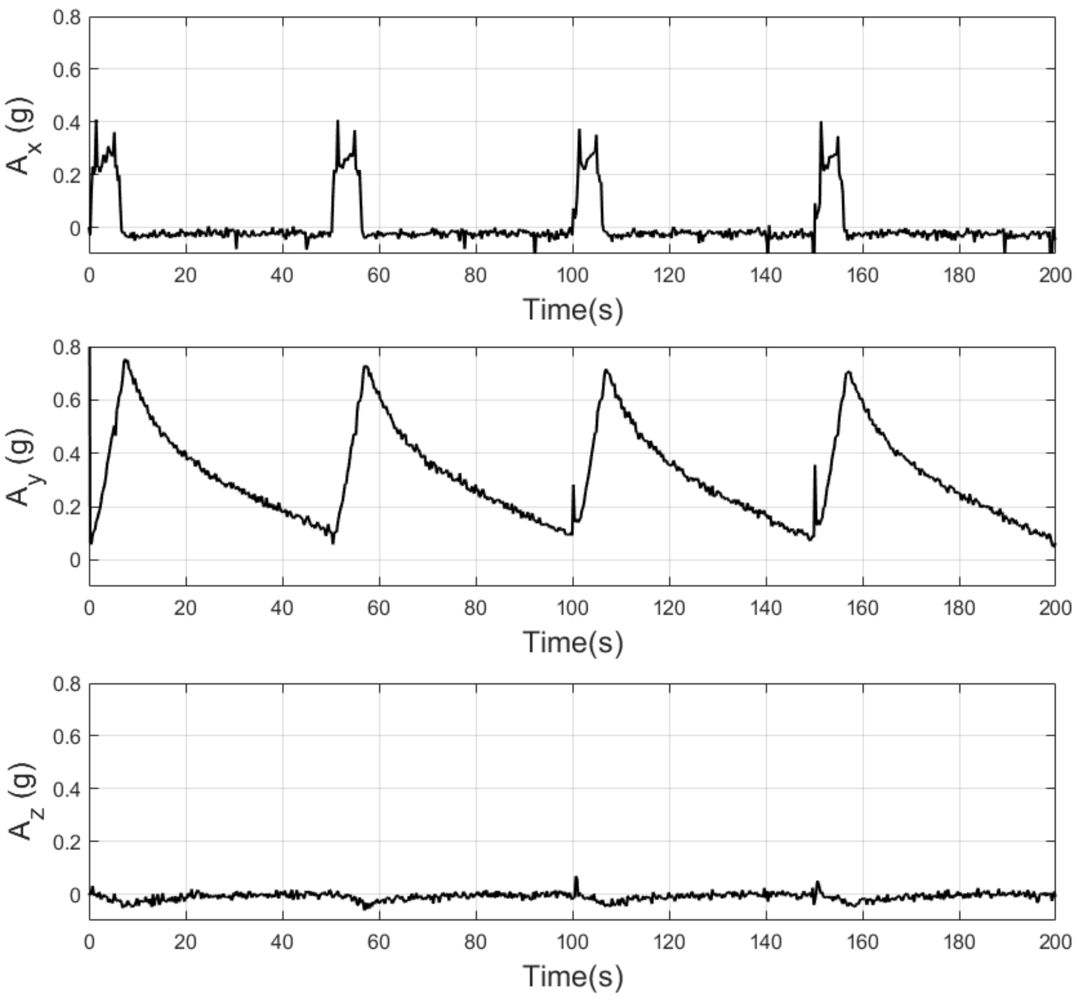


Figure 6.2: Ground truth accelerations from the Noisy Step Steer Data Set.

In order to test the robustness of the Gaussian radial basis function neural network to noise in the input and target measurements, Gaussian white noise was added to the step steer data set. Figure 6.1 shows the input and target time series for the noisy step steer dataset. The steer angle was stepped up several times throughout the course of the routine. This was done to increase the area of the input space spanned by the dataset. Noise and bias are major problems for inertial navigation because the position solution tends to drift over time once integrated. Because of this, dead-reckoned position solutions degrade over time, and hence a direct measurement update, from a GPS or otherwise, is required in order to keep the desired path. However, noise is less of an issue when directly estimating the acceleration output by the inertial measurement unit. During high-dynamic maneuvers, the accelerations in the longitudinal and lateral directions are generally large enough that the noise is of negligible importance. It is only in the vertical direction, in which the dynamics are the least observable, that the noise has a noticeable effect.

6.1 IMU Emulation

For each simulated data set, the neural network was trained to predict each of the target linear accelerations given the simulated inputs. The resulting estimated time-series are plotted against the truth values, along with the residual error from the estimation.

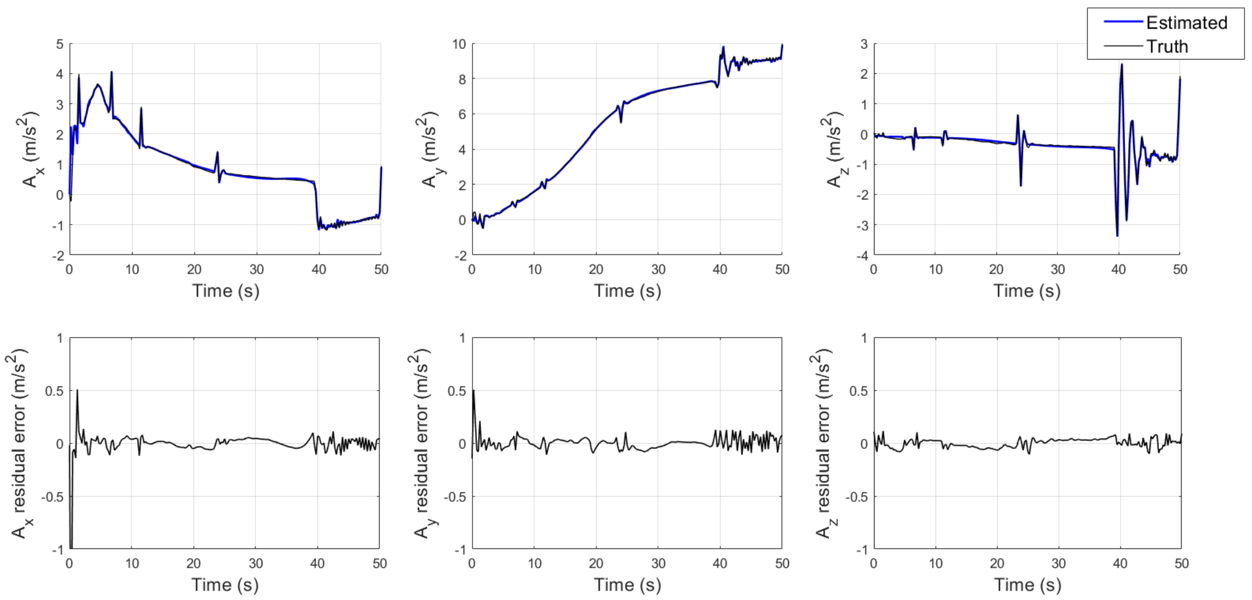


Figure 6.3: Above: Truth (black) vs Estimated (blue) linear accelerations for the 'Rough' Data set. Below: Residual error in the estimation.

The Rough data set tests the neural network's ability to generalize from training data collected on a smooth road surface to testing data with an identical maneuver collected on a rough road surface. This ability could have some implications for the potential to train the system fully on simulated data, without having to collect experimental data for every platform on which the method will be employed. Figure 6.3 compares the accelerations estimated by the neural network model with the measured accelerations. The residual error is shown below. Unsurprisingly, the GRBFNN was able to estimate the test accelerations with near perfect accuracy. The residual errors were approximately zero-mean and generally at or below $0.1\frac{m}{s^2}$

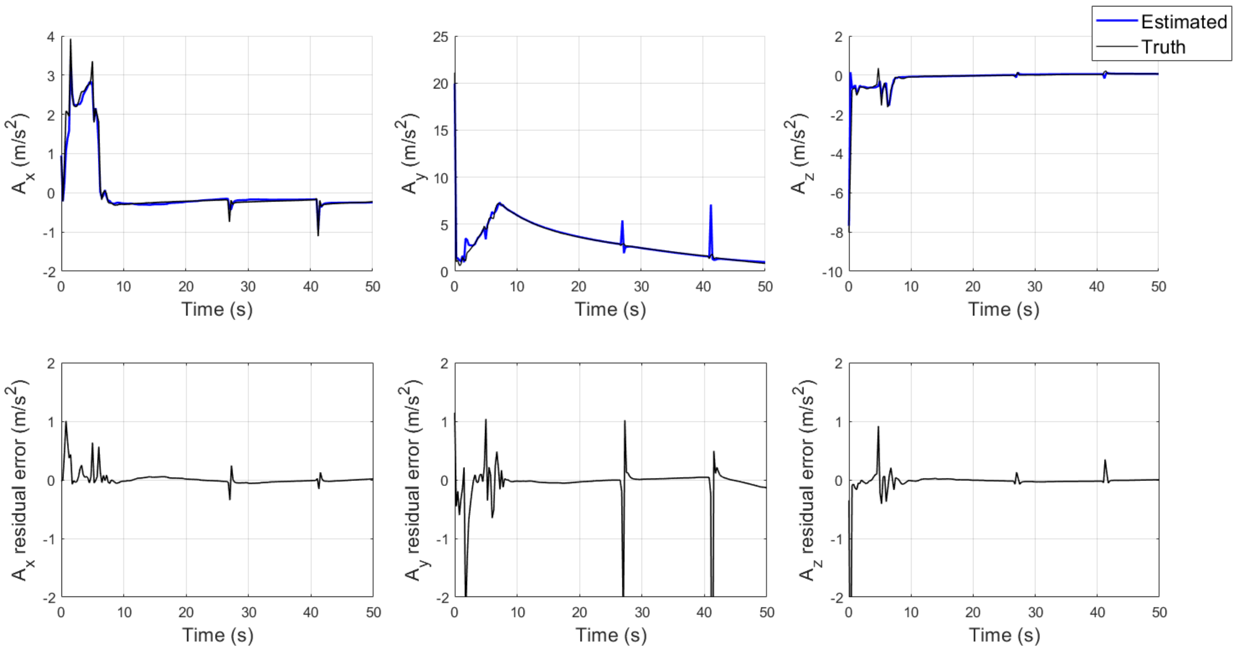


Figure 6.4: Above: Truth (black) vs Estimated (blue) linear accelerations for the Step Steer Data set. Below: Residual error in the estimation.

The Step Steer dataset was designed to assess the algorithm's ability to generalize to data sets with similar maneuvers, but different steer angle profiles. Figure 6.4 shows how the model performed on the step steer dataset. The GRBFNN was able to capture the general pattern in the accelerations. The neural network failed to correctly model the sudden peaks in acceleration visible in the training data set.

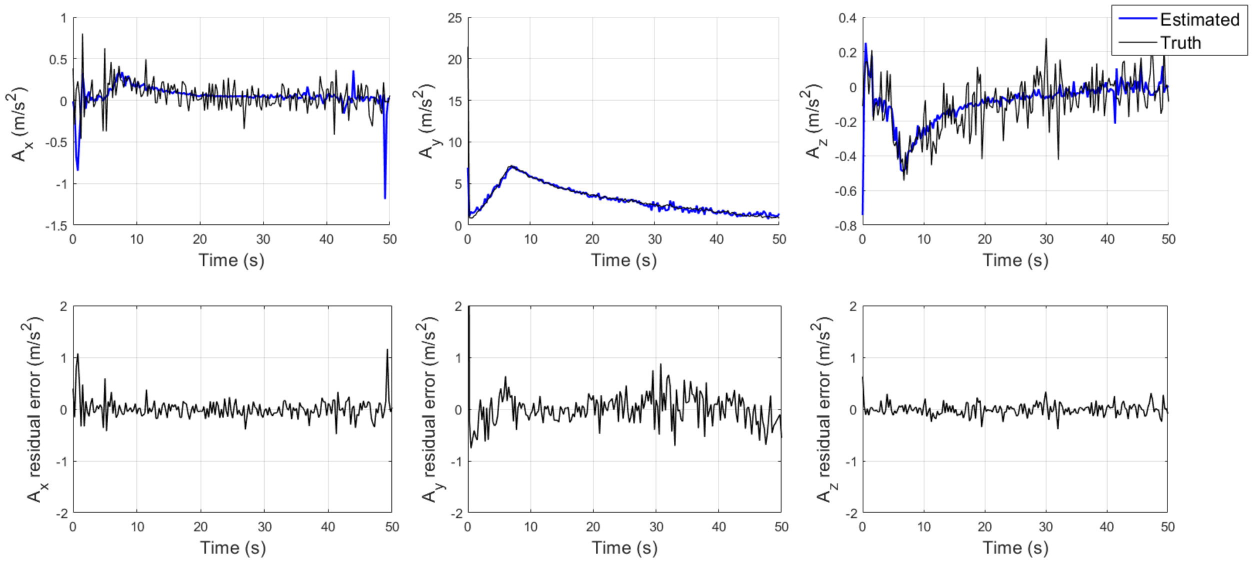


Figure 6.5: Above: Truth (black) vs Estimated (blue) linear accelerations for the Noisy Step Steer Data set. Below: Residual error in the estimation.

The GRBFNN also shows a remarkable ability to reject noise. Despite a fair degree of noise in the longitudinal and vertical directions, the acceleration estimate always tracks within the measurement error bounds, and is always close to the true value of the acceleration, despite being trained on noisy data. This result is demonstrated by Figure 6.5. While the measured accelerations, as well as the inputs the model was trained on, were effected by a significant degree of noise, the neural network was able to reject the noise and predict a smooth acceleration that is close to the undisturbed signal. This useful property arises out of the training procedure for the GRBFNN: basis functions are added one after another in sequence, and each basis function is optimized for every point in the training data set. Hence, only the general trends in the training data are captured, while local patterns are ignored. For smooth data, such as measured by any physical sensor, this is an ideal property.

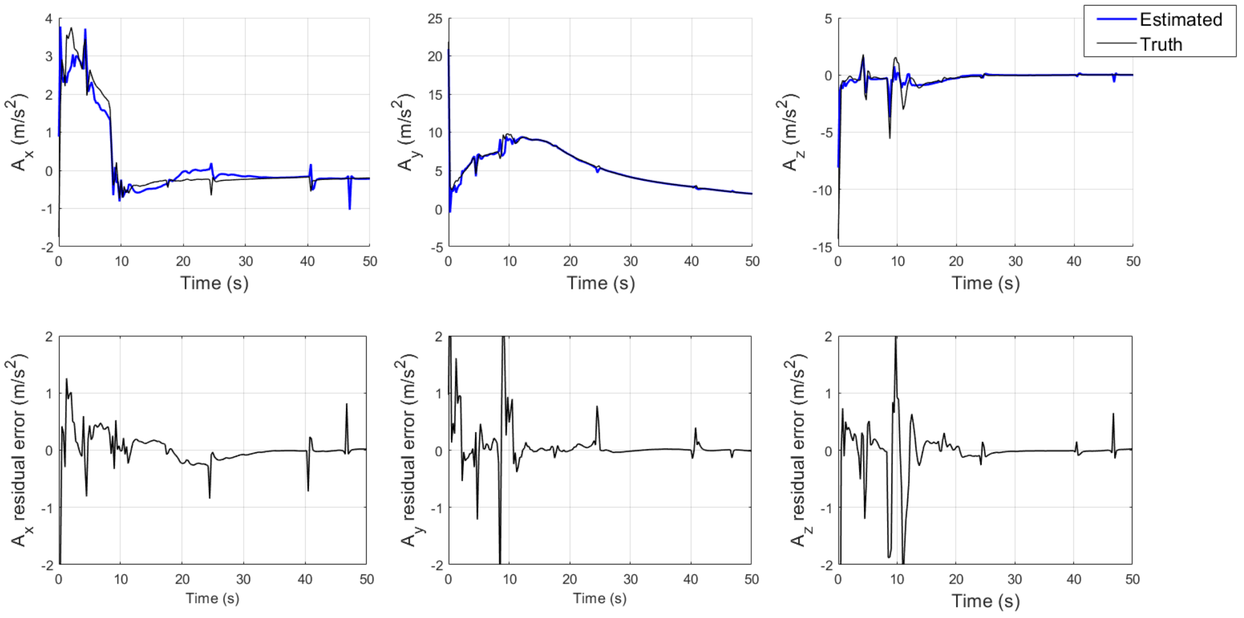


Figure 6.6: Above: Truth (black) vs Estimated (blue) linear accelerations for the Velocity Difference Data set. Below: Residual error in the estimation.

The neural network can also generalize to data sets with velocity profiles that were not seen during training. In the velocity difference test, the algorithm was trained for a set of low velocities and a set of higher velocities, and then tested on an unseen velocity of 40 km/hr. The steer profile of the maneuver was kept the same for all training and test runs. Figure 6.6 shows how the model performed on the velocity difference dataset. While the performance is generally worse for this data set than for the other simulated data sets, this was expected, given the high dynamics, and the high relative importance of the forward velocity to the measured linear accelerations. In general, however, the GRBFNN captures the dynamics well enough that there is no bias in the acceleration estimates.

6.2 Pose Estimation

For each of the simulated datasets examined above, the IMU emulation results were passed into the maximum likelihood search algorithm to obtain a corresponding sensor pose estimate. A covariance matrix adaptation equipped genetic algorithm was implemented in Matlab to solve for the optimal pose given the estimated linear accelerations, and the linear accelerations measured by the simulated sensor placed at an arbitrary location and with an arbitrary orientation on the vehicle's chassis.

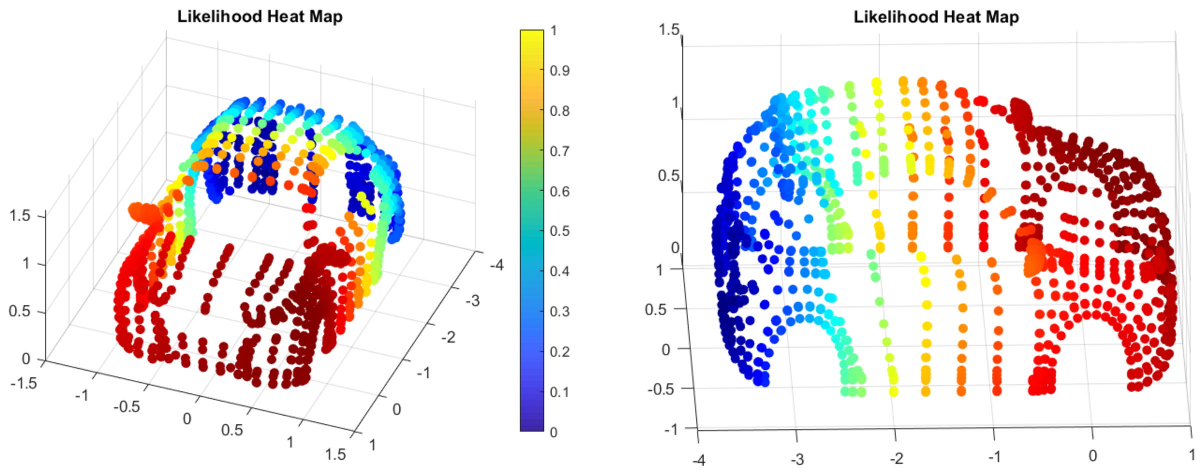


Figure 6.7: Likelihood Heat Maps for the C-Class Hatchback Mesh obtained from Carsim.

In each test the IMU was placed at a location near the front left wheel rim of the vehicle. Figure 6.7 shows how the likelihood varies across the body of the vehicle. The likelihood varies most strongly in the x-direction and less strongly in the y and z directions. This behavior is expected as most of the vehicle's dynamics took place in the longitudinal direction for this training dataset. The likelihood heat map also shows that the likelihood varies proportionally with distance from the true location of the sensor. Hence, the likelihood acts as a statistical sorting function for candidate points. The maximum likelihood therefore acts as a statistical estimator of the true sensor pose.

	Position Error			Orientation Error		
	X(cm)	Y(cm)	Z(cm)	$\Theta(deg)$	$\Phi(deg)$	$\Psi(deg)$
Rough	0.001	0.001	0.004	0.002	0.003	0.001
Step	0.150	0.010	0.004	0.002	0.003	0.001
Step + Noise	0.020	0.050	1.200	0.003	0.003	0.018
Velocity Difference	0.050	0.001	0.005	0.002	0.003	0.001

Table 6.1: Pose estimation error for each simulated dataset.

Table 6.1 shows the error for each component of the IMU-to-Vehicle pose for each of the four simulated datasets. For every dataset, the error in position was less than 2 mm in any direction, and the error in yaw angle was less than 0.1 degrees. The initialization routine produced a perfectly accurate result for the roll and pitch angles in every case. These results indicate that the neural network is capable of emulating the linear accelerations of an IMU to a strong enough degree to enable mm level precision, and that the maximum likelihood search method is robust to errors in the emulation when they do occur.

Chapter 7

Experimental Results

The direct IMU-to-Vehicle extrinsic calibration method detailed in previous chapters was subjected to a battery of experimental trials. These trials were designed both to collect experimental data with which to train the IMU emulation model, as well as to analyze the performance of the algorithm on a real-world platform. While the viability of the algorithm has already been demonstrated in simulation, experimental validation is crucial to show that it can perform to a similar standard in practice. Real-world platforms are subject to measurement error due to noise and bias, as well as perturbations and nonlinear variability in the dynamic model due to effects that cannot be reliably modelled in simulation. Dealing with these effects requires special analysis, and a modified approach to solving the modelling and estimation problem.

7.1 Experimental Setup

All of the experimental trials were conducted on the same platform, equipped with the same set of sensors. The Dataspeed 2017 Lincoln MKZ shown in Figure 7.1 served as the platform for all of the experimental tests. This vehicle is similar in design to the average commercial surface-road vehicle, so it should in theory have similar dynamics as well. To conduct each trial, a minimal set of sensors was fixed to the body of the MKZ, in addition to a ground truth system consisting of a high-grade IMU and a GPS receiver. The ground truth system combines GPS position fixes with accelerations and angular rates to provide a fused position solution that yields velocities, angular velocities, and accelerations in the vehicle frame. Hence the target accelerations in the vehicle frame can be obtained, as well as the corresponding input vectors, that are needed to train the IMU emulation model.

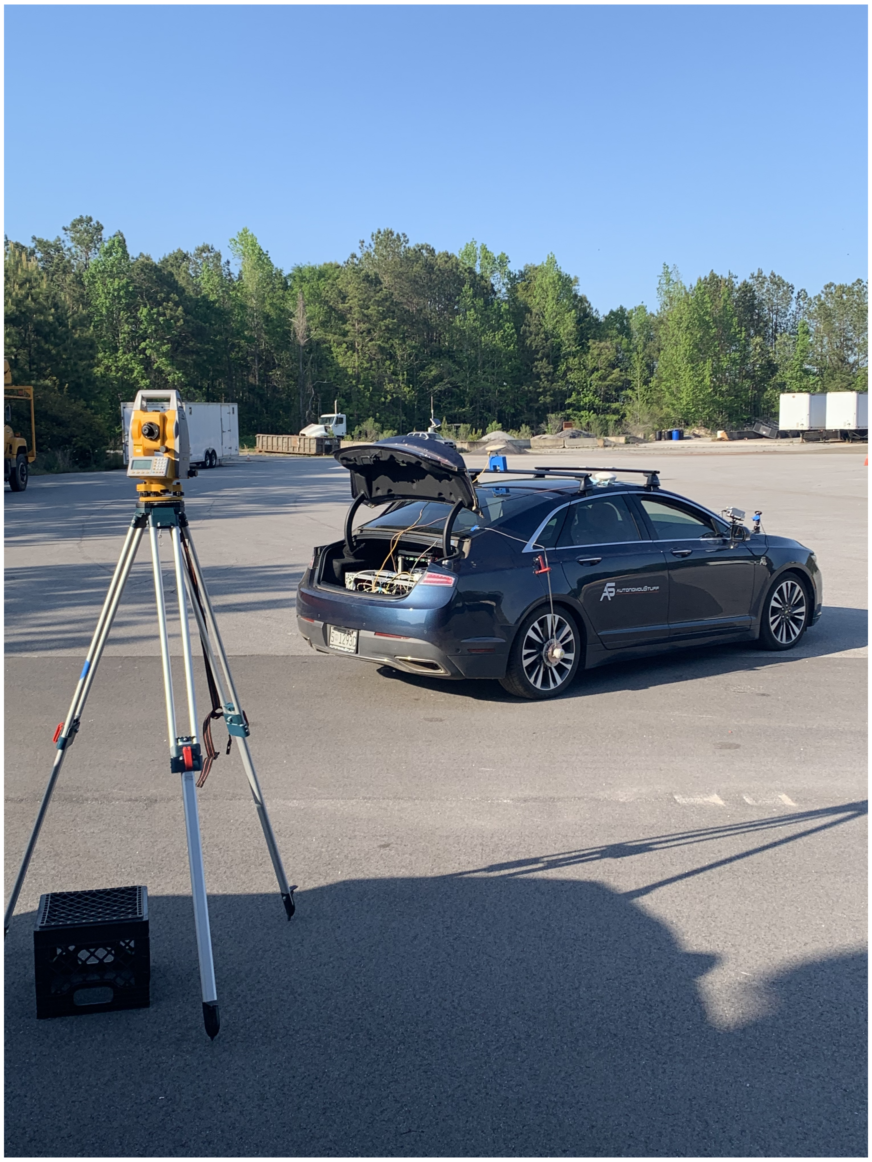


Figure 7.1: Experimental setup. The surveying tool (at left) was used to capture the ground truth sensor positions and orientations.

The vehicle was also equipped with two GPS antennas, and the IMU which was to be calibrated. The first antenna was connected to a Novatel receiver, while the second was connected to a Septentrio receiver. Since the objective of this research is to enable the dynamic configuration of the sensors on a self-driving car, it was not assumed that the GPS antennas would be aligned in a meaningful way. Hence they were placed at random locations on the roof of the MKZ. According to the same logic, the IMU was mounted at an arbitrary location on the front hood, and rotated to an arbitrary angle, such that the pitch, roll, and yaw of the sensor relative to the vehicle would all be non-negligible. This choice was intended to mimic the worst case scenario mounting, for which a fusion algorithm that relies on extrinsically calibrated measurements would likely fail. The IMU employed in all of the experimental trials was a Vectornav VN-100. The VN-100 is a small commercial grade unit that can easily be transported and mounted in different configurations [26].

The vehicle itself is also equipped with a steering wheel encoder, which measures the angle of the steering wheel at any given time. This element is crucial to the IMU emulation phase of the calibration procedure, because it provides reliable information about the state of the vehicle without requiring its own prior extrinsic calibration. It is also highly correlated with the lateral and longitudinal components of the vehicle's acceleration which makes it an invaluable input for the IMU emulation phase. The MKZ also contains a low-cost wheel odometer in each wheel. These are used to measure the vehicle's forward velocity, which is used to calculate one of the inputs to the IMU emulation neural network. The mounting of the GPS antenna and the VN-100 IMU are shown in Figure 7.2.

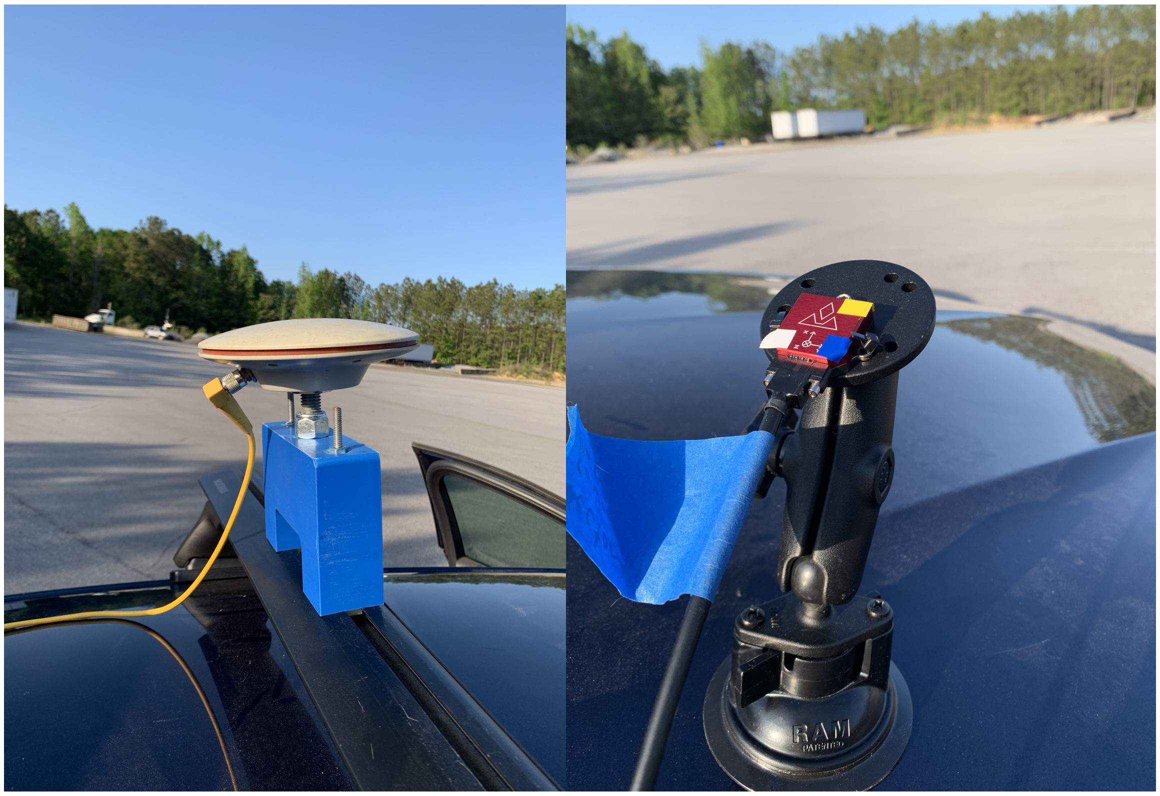


Figure 7.2: Sensors mounted on the vehicle body.

In order to verify the performance of the calibration procedure, the ground truth position and orientation of the IMU relative to the vehicle frame were measured using a Total Station laser surveying tool, shown at bottom left in Figure 7.1. The Total Station captures the locations of points in physical space within a spherical reference frame centered on the device. It measures the distance to a given target by emitting a series of laser pulses. The average return time of the pulses determines the distance between the device and the target. The polar and azimuthal angles of the target point are recorded by rotary encoders housed within the device. The Total Station can capture the locations of points in space more accurately than any other commercially-available device. It is accurate to within 1 mm in distance, and less than 0.1 degrees in each angle.

Figure 7.3 illustrates the process of collecting the ground truth sensor positions and orientations with the Total Station. Each sensor is pictured in approximately the same location as it was mounted for all of the experimental tests. The objective in collecting the ground truth sensor positions and orientations with the Total Station was to obtain accurate measurements of these values relative to the vehicle frame. However, the Total Station measures position and orientation in its own reference frame. Hence, the vehicle's position and orientation in the Total Station's reference frame needed to be captured as well. To do this, the centers of each of the four wheel hubs on the MKZ were measured. The center of the rear axle could then act as the origin of the vehicle frame, while the x and y axes would be formed by the vectors between the right and rear wheel pairs respectively. However, the four wheel centers of the vehicle are not all visible from a single survey position. A second survey position was needed in order to capture the wheel centers on the left side. These points would then need to be transformed from survey reference frame 2 into survey reference frame 1, in which all of the other points were measured.

The ground truth data captured by the Total Station is shown overlaid with a mesh of the Lincoln MKZ in Figure 7.4. This figure illustrates where each sensor was located during training and testing. The VN-100 IMU, in red, was mounted on the front right hood of the vehicle. The ground truth system, in green, was mounted in the back trunk. Three GPS antennas, meanwhile, were mounted on the roof, and are shown as separate points in yellow. The wheel centers were also captured to provide the ground truth vehicle frame orientation. Anchor points, shown in blue, were captured both on the ground, and on the vehicle to provide three dimensional observability of the transformation between survey position 1 and survey position 2.

The primary source of error in the ground truth positions and orientations was therefore not the measurement error of the Total Station, but rather human error in attempting to capture very specific points on each device. For each device mounted on the test vehicle, its location was measured by fixing the Total Station on the center of its top surface. For the

IMU and the ground truth system, the orientation was obtained by measuring three corners of the device. These corners then formed a set of orthonormal basis vectors that defined the orientation of the sensor's reference frame relative to the reference frame of the Total Station at the current survey position.

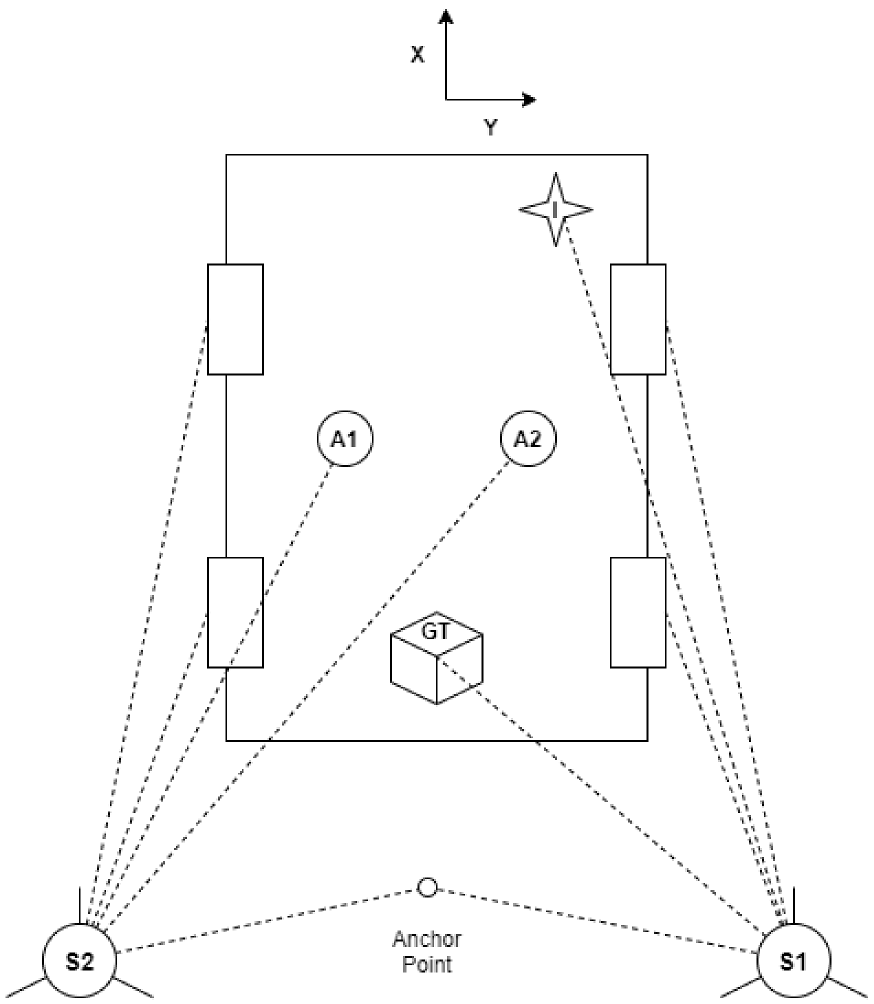


Figure 7.3: Ground truth lever arms and orientations were captured using a laser surveying tool. A bird's eye view shows the locations of the sensors on the vehicle and the two survey positions.

Transforming a set of points in 3D space from one coordinate frame into another requires a set of 'anchor points' shared by both. The minimum number of anchor points needed to fully specify the relative pose of the two reference frames is three. However, the error in the transformation is reduced by increasing the number of common points; it was empirically determined that 12 anchor points would be sufficient to estimate the transformation to approximately 0.1 degrees of error in each direction.

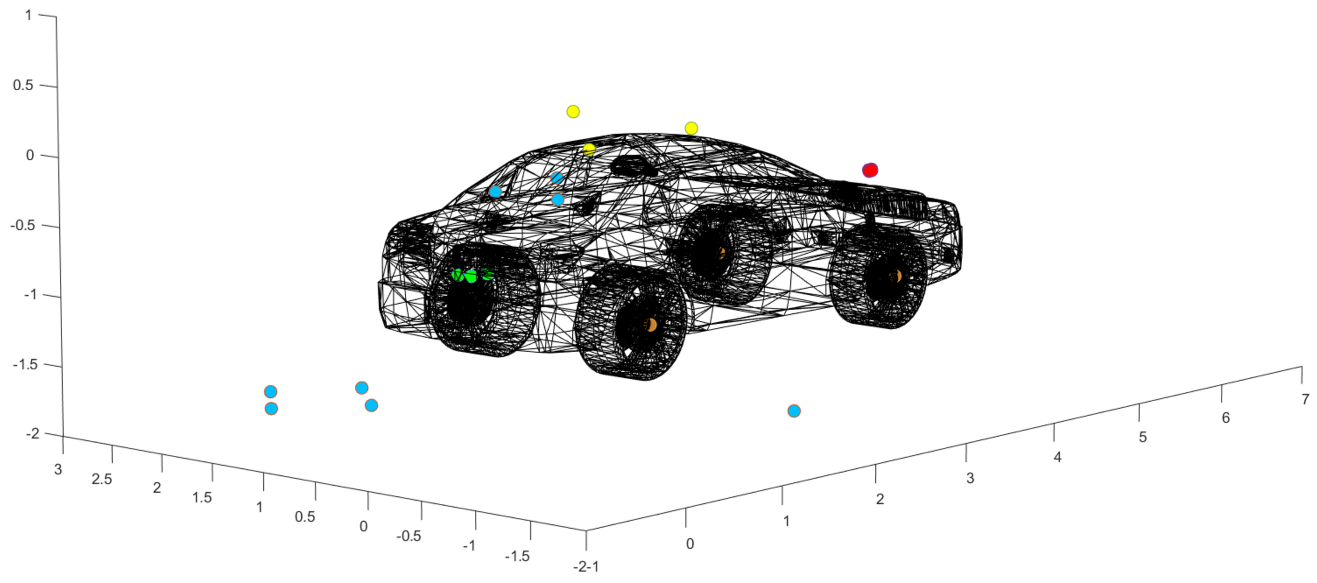


Figure 7.4: Ground truth sensor positions captured by the laser surveyor. Red: IMU, Yellow: GPS antennas. Green: Ground truth system. Bronze: Wheel centers. Blue: Anchor points.

Before the survey data could be parsed into ground truth Sensor-to-Vehicle poses, however, manual corrections needed to be made. Aligning the laser surveyor with the exact corners of either the Vectornav IMU, or the ground truth system, proved difficult. To ensure that the acquired sensor reference frames were properly orthonormal, the positions of the corners of each device were measured in relation to the points measured with the surveyor.

This was accomplished by manually measuring the distance between the measured points, corners, and a number of other feature points on the device. This created a matrix of measurements, whose error could then be minimized by an optimization algorithm. As in the maximum likelihood search method, the optimization was performed by a covariance matrix adaptation genetic algorithm.

After corrections were made, the points were then transformed into survey reference frame 1, and from there into the vehicle frame. Once this had been accomplished, the Sensor-to-Vehicle orientations were extracted using a Wahba formulation of the attitude determination problem. Wahba's problem seeks to find the rotation matrix that minimizes the least squares error between a set of points, and another rotated set of points [102]. This minimization problem can be written mathematically as in the equation below.

$$J(\mathbf{R}) = \frac{1}{2}\sum_{i=1}^{N} w_i \left\| \vec{v}_{i1} - \mathbf{R}\vec{v}_{i2} \right\| \tag{7.1}$$

In 1988, a solution to Wahba's problem was presented, which used the singular value decomposition [68]. A matrix B is formed by taking the matrix dot product between the set of vectors formed from points in reference Frame 1, and the set of vectors in reference Frame 2.

$$\mathbf{B} = \sum_{i=1}^{N} \vec{v}_{i1}\vec{v}_{i2}^{T} \tag{7.2}$$

The singular value decomposition of B is then taken, and a matrix M is formed from the determinants of its left and right eigenvector matrices.

$$\mathbf{B} = \mathbf{U}\mathbf{S}\mathbf{V}^{T} \tag{7.3}$$

$$\mathbf{M} = diag([1\,1\,det(\mathbf{U}) * det(\mathbf{V})] \tag{7.4}$$

Finally, the optimal rotation matrix, R, is calculated.

$$\mathbf{R} = \mathbf{UMV}^T \tag{7.5}$$

Using this formula, the ground truth orientations of the IMU relative to the vehicle frame, as well as those of the ground truth system, were obtained.

After the ground truth survey data had been collected, a series of experimental data collection trials were conducted. A figure-8, square, road drive, and combined calibration maneuver were performed. These maneuvers were chosen for a combination of observability and ease of execution. Together they provided both an exploration of the dynamics sufficient to train the IMU emulation model, and a reasonable test case for the calibration procedure.

7.2 Figure-8 Trial

A dataset composed of successive figure-8 maneuvers was chosen as the primary dataset, both for training the neural network IMU emulation model, as well as for analysis of the calibration procedure. A figure-8 maneuver excites the vehicle dynamics to a strong degree, and therefore provides high observability of the IMU-to-Vehicle lever arm and angular misalignment. Because it explores the nonlinear dynamics of the vehicle to a large extent, the figure-8 also serves as a perfect medium to train the IMU emulation model. It would also be an ideal maneuver for the calibration procedure, but it is less feasible than other maneuvers, such as the square maneuver, because it requires a large open space to execute.

7.2.1 Filtering

While analyzing the data collected for the experimental trials, a high degree of vibrational noise was noticed in the accelerations and angular rates of both the Vectornav IMU, and the ground truth system. This noise was likely due to vibration of the motor, and a lack of dampening on the mounted devices. The amplitude of the noise was sufficiently large in the vertical direction that it overwhelmed the signal of the vertical acceleration. As a consequence, the neural network was not able to learn the target vertical acceleration from the measured inputs. Without an effective estimate of the vertical acceleration, the pose estimate obtained was necessarily degraded. The pitch and roll rates also experienced an overwhelming degree of vibrational noise, which had an even greater negative effect on the pose estimate. The pose estimation phase is highly dependent on accurate measurements of the angular rates and their derivatives, the angular accelerations, which form the rigid-body transformation matrix.

A set of low-pass filters was designed to address the issue. Zero pass band attenuation and zero phase distortion were important properties sought in the filter. These features were crucial to ensure that the vehicle dynamics were preserved intact through the filtering process as best as possible. The form of the filter was therefore chosen to be a digital infinite impulse response smoothing filter with a forward and a backward pass. Taking advantage of the fact that the filtering does not need to be performed in real time, a backward pass through the data is performed, which corrects the phase lag and distortion caused by the forward pass. The matlab function 'filtfilt' was used to generate the filter to the chosen specifications.

The impact of the vibrational noise is visible in the Fourier decomposition of the measured properties. Figure 7.5 shows peaks at approximately 17, 26, and 35 Hz, indicating the effect of high-frequency vibration on the measured accelerations. Large peaks at 1.7 Hz and 2.4 Hz, meanwhile, indicate the presence of low-frequency vibrational modes as well. While the former is relatively easy to isolate from the vehicle dynamics, the latter is not, and is therefore much more difficult to deal with. The results shown at bottom demonstrate that the filtering had the intended effect. The major noise peaks were all removed, which is especially noticeable for the vertical acceleration. However, it demonstrates that some of the low-frequency noise in the vertical acceleration could not be removed by the filter.

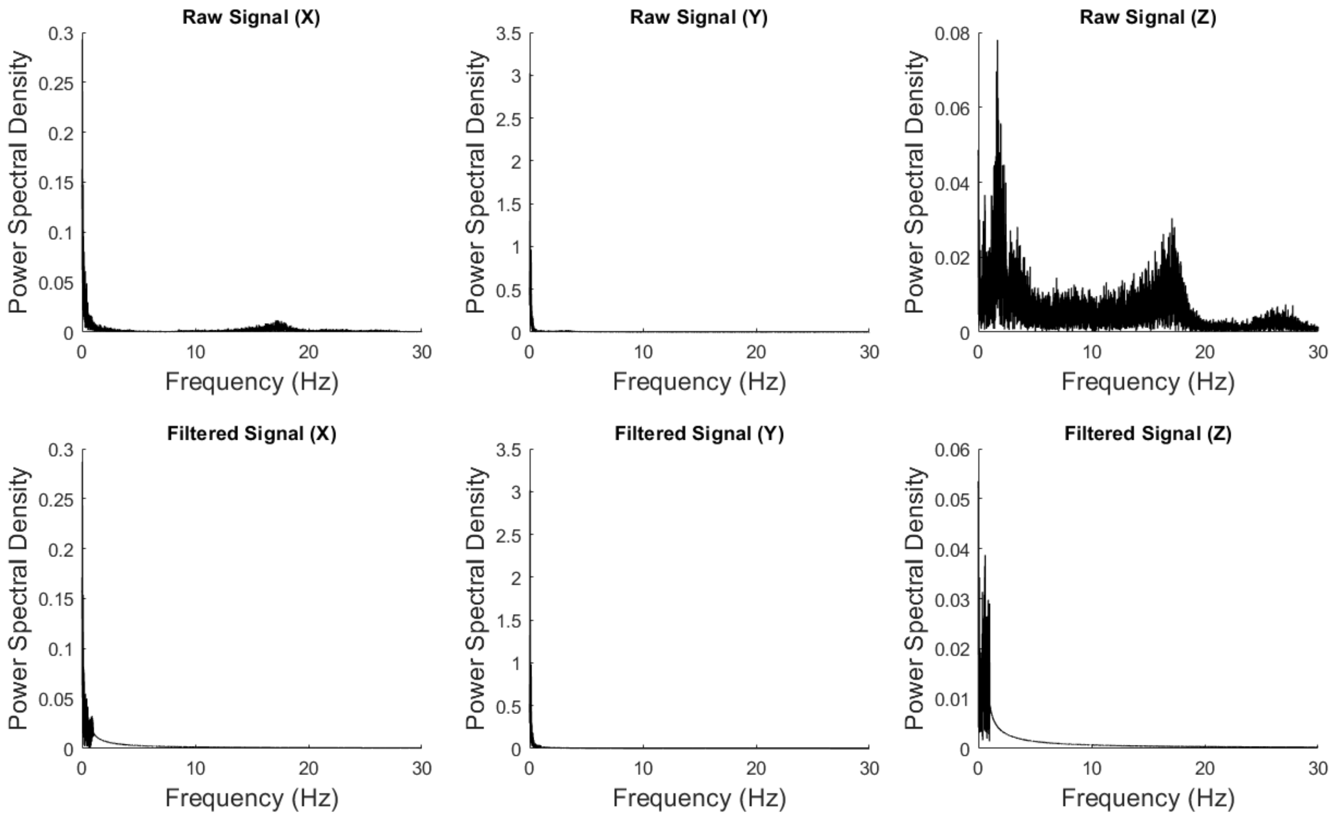


Figure 7.5: Spectral decomposition of the ground truth accelerations, before filtering (Top) and after filtering (Bottom).

The same vibrational modes are present in the angular rates measured by the ground truth system, as well as the accelerations measured by the IMU mounted on the body frame of the vehicle. For the angular rates displayed in Figure 7.6, the vibrational noise is most disruptive in the pitch and roll directions. This is due to the low observability of the pitch and roll dynamics on flat ground. Obtaining accurate angular rates is of particular importance, because the rigid-body transformation matrix is formed from the angular rates and their derivatives, the angular accelerations. A slight inaccuracy in the angular rates can lead to a correspondingly large error in the estimated sensor position. This is doubly true, because the angular accelerations are derived from the angular rates, rather than being directly measured. This means that any error in the angular rates is amplified in the angular accelerations, and consequently in the sensor position estimate. While low-pass filtering eliminates the bulk of the noise due to high-frequency vibration, low-frequency modes cannot be eliminated without removing some of the dynamics. In this case, the only viable option is to give the true signal time to accumulate power and overwhelm the noise.

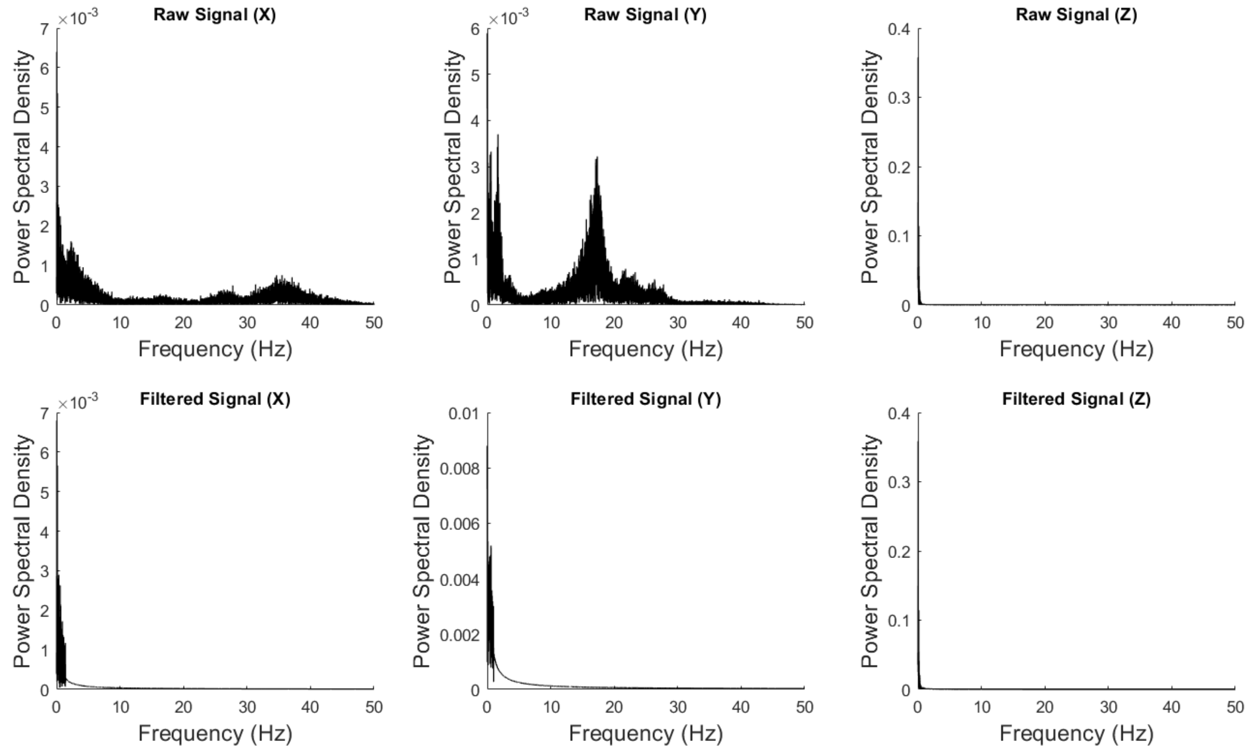


Figure 7.6: Spectral decomposition of the ground truth angular rates, before and after filtering.

The IMU-to-be-calibrated's linear accelerations underwent similar disturbances, and hence were filtered in the same way as the ground truth accelerations and angular rates. Figure 7.7 shows how the sensor accelerations were filtered. The IMU has a lower sample rate than the ground truth system, so the high frequency vibrational modes are not visible in this plot. Fortunately, the low-frequency modes detected in the ground truth measurements are not present in the spectral decomposition of the IMU's linear accelerations. The only significant contributor to error is the white noise in the vertical acceleration. A bandwidth of 1 Hz was therefore chosen for these signals.

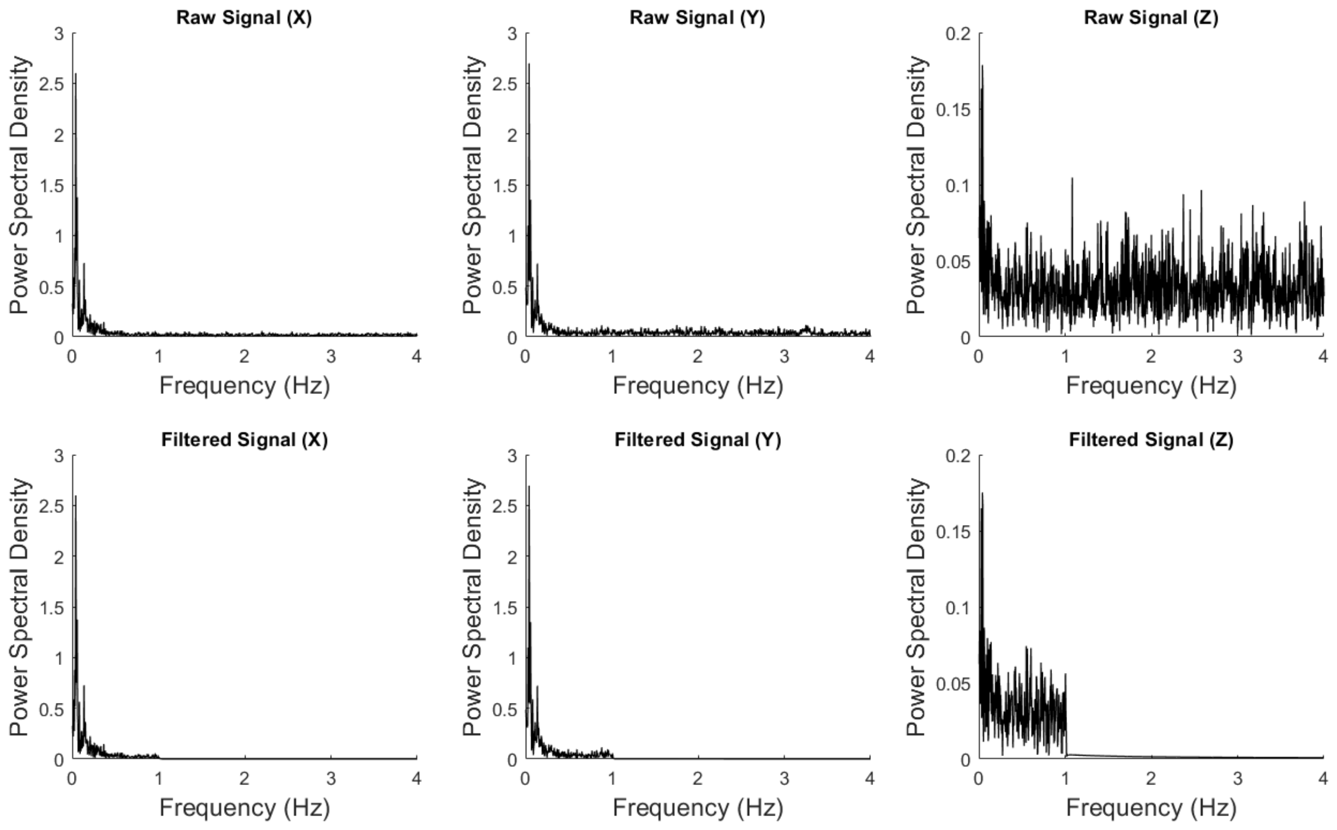


Figure 7.7: Spectral decomposition of the measured sensor accelerations, before (Top) and after (Bottom) 1Hz low-pass filtering.

The resulting filtered signals are compared with the raw signals in Figures 7.8 - 7.10. The difference between the raw and filtered signals is most drastic in the vertical direction, where the vibrational noise was strongest. However, the filtering process is still important for the longitudinal and lateral directions, because an accurate IMU position estimate is highly dependent on accurate acceleration estimates.

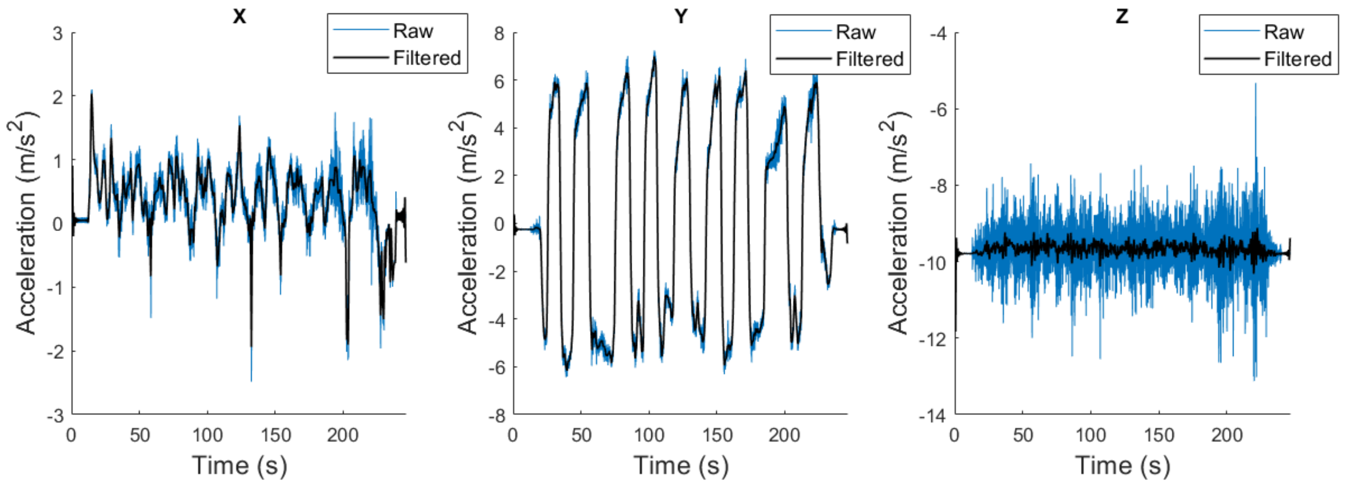


Figure 7.8: Raw ground truth accelerations compared to filtered ground truth accelerations for the Figure-8 dataset.

Figure 7.8 compares the filtered ground truth accelerations with the raw ground truth accelerations

Figure 7.9 compares the filtered ground truth angular velocities to the raw angular velocities. The filtering process makes little difference in the yaw, because the dynamics are much stronger around the z-axis. However, the pitch and roll angular rates are necessary to obtain a six degree of freedom pose estimate. A substantial degree of noise is removed from these measurements, which will ensure a more accurate rigid-body transformation matrix, and therefore a more accurate pose estimate.

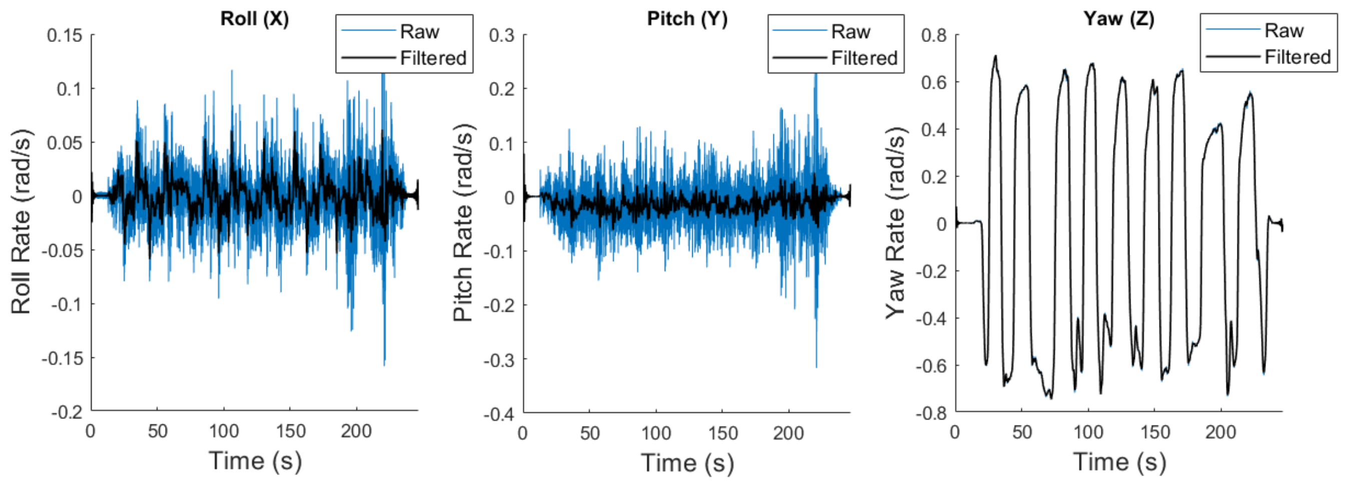


Figure 7.9: Raw ground truth angular rates compared to filtered ground truth angular rates for the Figure-8 dataset.

The raw measured accelerations in the vehicle frame are shown in Figure 7.10. Filtering was particularly crucial for the measured accelerations, because the Vectornav IMU is lower quality than the ground truth system, and therefore the magnitude of the noise that affects it is higher. The experimental tests demonstrated that the error in the pose estimate is roughly proportional to the standard deviation of the error. Assuming a perfect model of the vehicle frame accelerations, the noise in the sensor measurements would still cause some degree of pose estimation error. The accuracy of the of the pose estimate is consequently dependent on the quality of the IMU-to-be-calibrated. Filtering the IMU measurements first eliminates some of that inherent error.

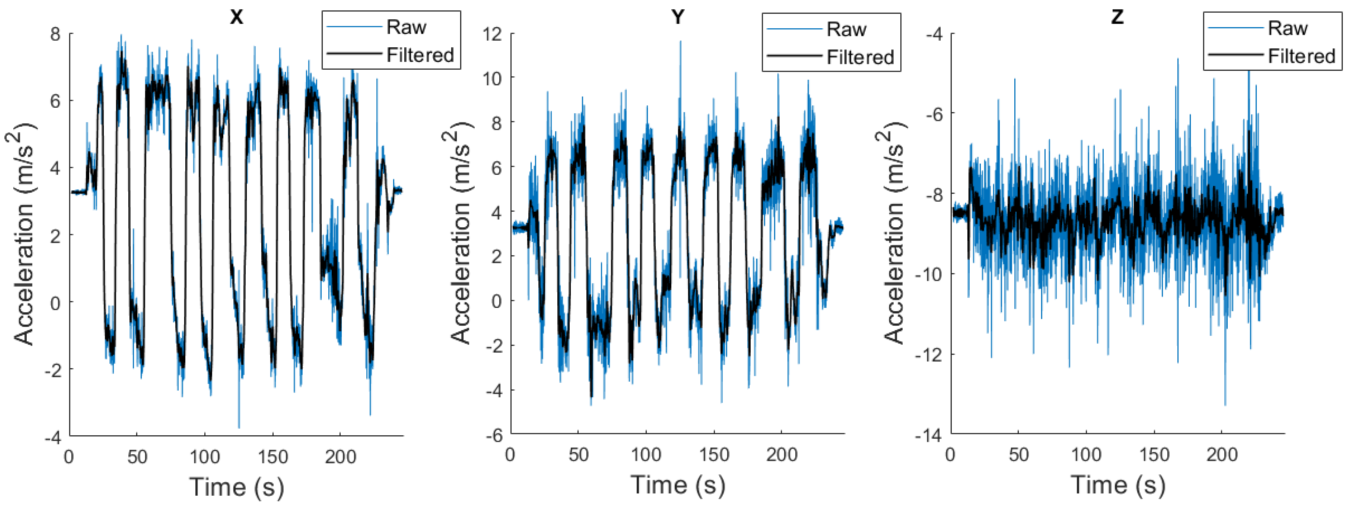


Figure 7.10: Raw measured sensor accelerations compared to filtered sensor accelerations for the Figure-8 dataset.

Filtering the measured accelerations and angular rates of the ground truth system during training, and the VN-100 IMU during testing, was observed to substantially improve the performance of both the IMU emulation phase, and the pose estimation phase. Given the low observability of the vertical component of acceleration, as well as the pitch and roll rates, filtering was crucial in eliminating the influence of sensor noise on the estimated lever arm in particular. However, if the bandwidth is set too low, this may destroy information about the vehicle dynamics. Hence, it is important to select the correct filter parameters for the system, and the sensors being used.

7.2.2 IMU Emulation

The GRBFNN IMU emulation model was trained on a sequence of figure-8 maneuvers conducted on a concrete skid pad. This maneuver was expected to both maximize the observability of the IMU-to-Vehicle lever arm, and to span the input feature space to a sufficient degree. A new set of input features was introduced to the model to improve its performance on experimental data. While obtaining accurate velocities and angular rates in the vehicle frame is easy in simulation, it is much more difficult in practice. The IMU emulation model also needs to be more accurate when applied to experimental data, in order to obtain a similar degree of accuracy in the pose estimate as in simulation. The measured angular rates are subject to both white noise, and vibrational noise, as discussed in the previous section. This noise corrupts the rigid-body transformation matrix, which in turn disrupts the pose estimation process.

Figure **??** shows the raw input time series collected for the figure-8 dataset. The steer angle was kept to roughly the same pattern throughout the run, though there were some minor differences from maneuver to maneuver. These differences allow the input space to be explored more fully. It is important to span the input space to as great an extent as possible during training, so that the dynamics learned by the neural network model can be recalled during testing. The figure-8 is an optimal maneuver for this purpose. However, the forward velocity was kept around 6-10 m/s throughout the run, which limited the region of the input space that could be traversed. This was a limitation of the data collection procedure; the Lincoln MKZ was driven around two cones placed at a fixed distance. Ideally, the neural network model would be trained with a set of figure-8 maneuvers with varying turning radii and forward velocities. A training procedure of this sort could potentially span the entire operating range of the input feature space. However, the available open space was not large enough to accommodate this for the experimental tests.

The set of input features used for all of the experimental tests in this thesis included the normalized steer angle, the magnitude of the velocity, and a proxy for the lateral slip ratio. While the lateral velocity cannot be directly obtained from the sensor measurements that are available on the vehicle, the lateral slip ratio can be approximated from the magnitude of velocity, and the forward velocity of the vehicle, assuming that the vertical component of the velocity is negligible. After considerable experimentation, it was concluded that this set of features best captured the vehicle's dynamics.

Figure 7.11 shows the normalized input features for the figure-8 dataset. The steer angle was scaled by a factor of 1000 to improve the training dynamics. The velocity magnitude was calculated from its individual components using the Pythagorean theorem. The lateral slip ratio was then estimated from the velocity magnitude and the forward velocity using the following approximation:

$$Lateral\ Slip\ Ratio\ =\ \frac{v_{v,y}}{v_{v,x}} \approx (\frac{V_v - v_{v,x}}{v_{v,x}}) \tag{7.6}$$

where V_v is the magnitude of the vehicle velocity measured by GPS and $v_{v,x}$ is the vehicle forward velocity measured via wheel odometry. The lateral slip ratio is maximal during high dynamic maneuvers, in particular during sharp turns. When the vehicle is travelling straight, as it does when transitioning from one section of the figure-8 to the next, the lateral slip is zero. This explains the pattern seen in the normalized inputs for the figure-8 dataset. The addition of the lateral slip ratio to the set of input features allows the GRBFNN to model accelerations better under higher dynamic regimes. It also allows the vehicle dynamics to be modelled for a range of different surface conditions. This could be particularly useful if, for example, the calibration routine is performed when the road surface is slick due to ice or rain.

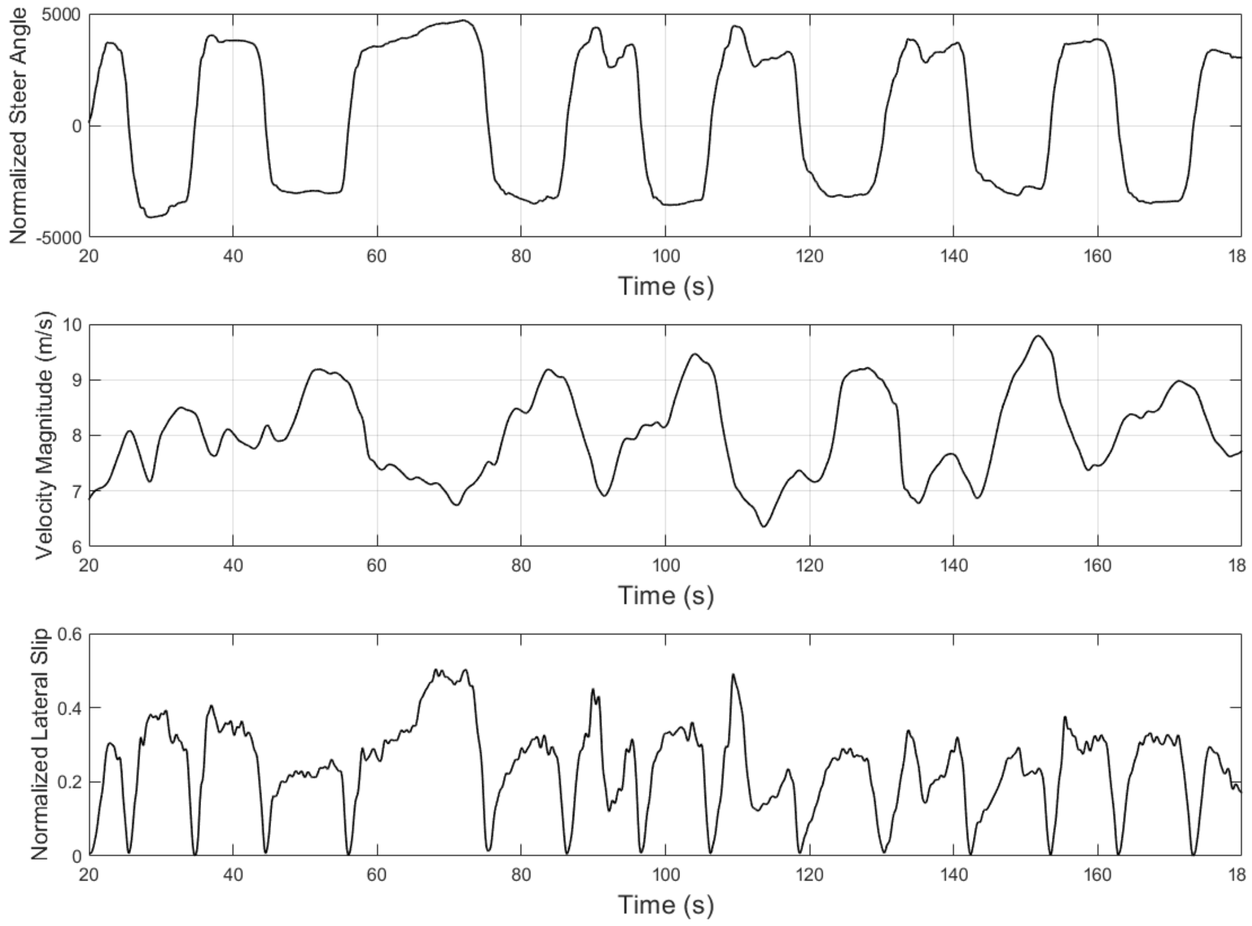


Figure 7.11: Normalized input time series for the Figure-8 dataset.

The ground truth accelerations for the figure-8 dataset are shown in Figure 7.12. The dynamics are most observable in the lateral direction due to the high steer angle through the turns. Positive and negative spikes in the longitudinal acceleration indicate application of the throttle or brake respectively. As expected, the dynamics are much weaker in the vertical direction. However, there is still a cyclical pattern that matches with the cyclical pattern of the figure-8 maneuver.

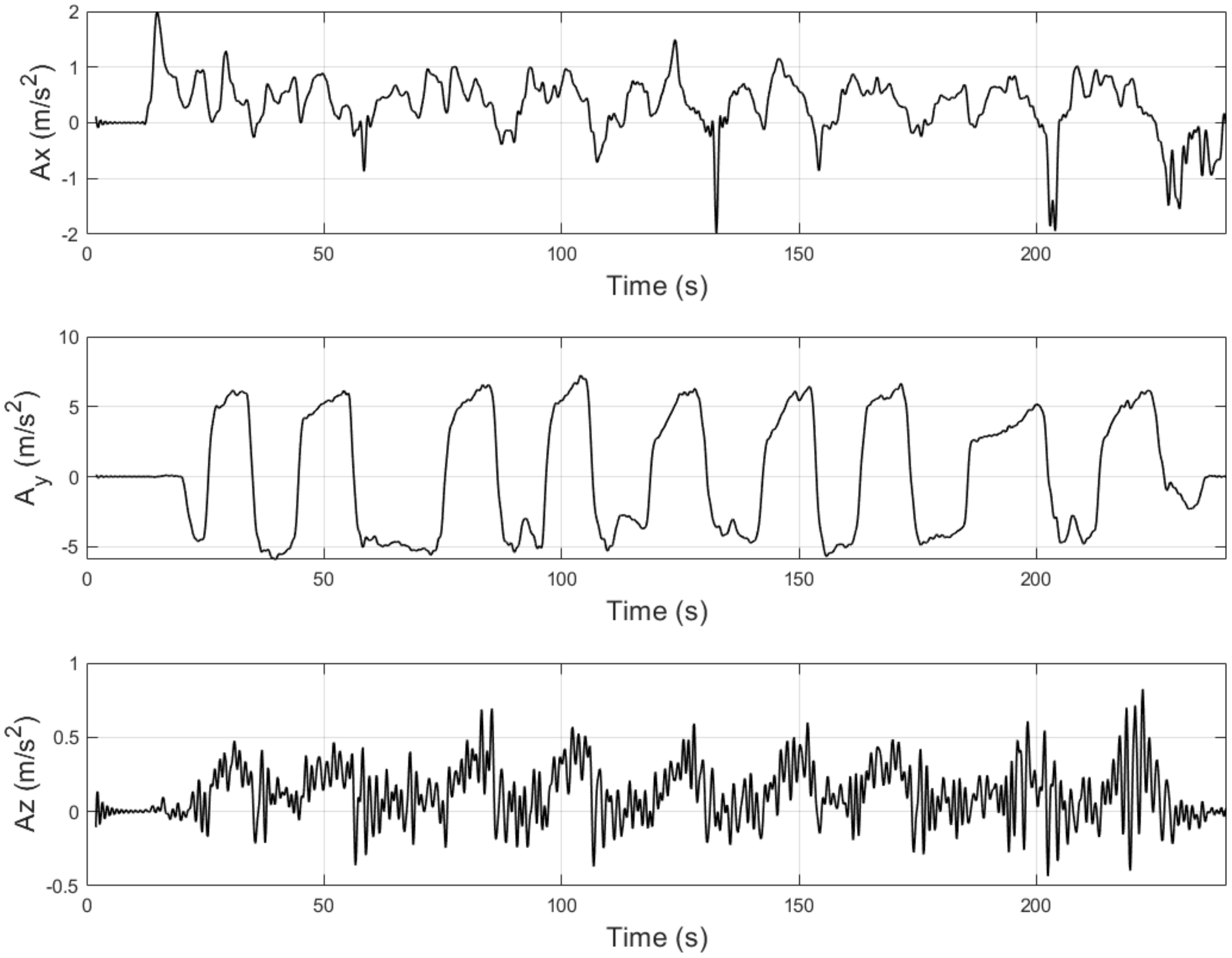


Figure 7.12: Target accelerations for the Figure-8 dataset.

Figure 7.13 depicts the traversal of the training dataset through the input feature space, colored according to the target nonlinear lateral acceleration. The left-hand plot shows how the training routine varies from maneuver to maneuver, covering different regions of the input space. The dark blue and bright yellow clusters represent the height of each left and right turn respectively, which corresponds to the maximum negative or positive lateral acceleration. The right-hand plot shows how the magnitude of the lateral acceleration is directly proportional to the lateral slip. A quadratic relationship exists between the lateral slip ratio and the steer angle. At large steer angles, the lateral slip is high, and so is the target acceleration.

An important property of the chosen set of features is that it separates input vectors with different target accelerations, such that the target function varies smoothly within the input feature space. However, if the dimensionality of the input space is reduced too much, then the same input vector may be mapped to multiple different values of the target function. The neural network will only learn one of these possible values during training. At test time, the model will only be able to recall this value, which will very likely be incorrect. Choosing the correct input space is therefore crucial, because a wrong choice can result in systematic model error. The set of features used in this thesis was chosen by analyzing the statistical dependence of the individual features. Statistically independent features provide additional information that enables the neural network to learn the dynamics more precisely, while statistically dependent features are redundant and unnecessarily increase the dimensionality of the input space.

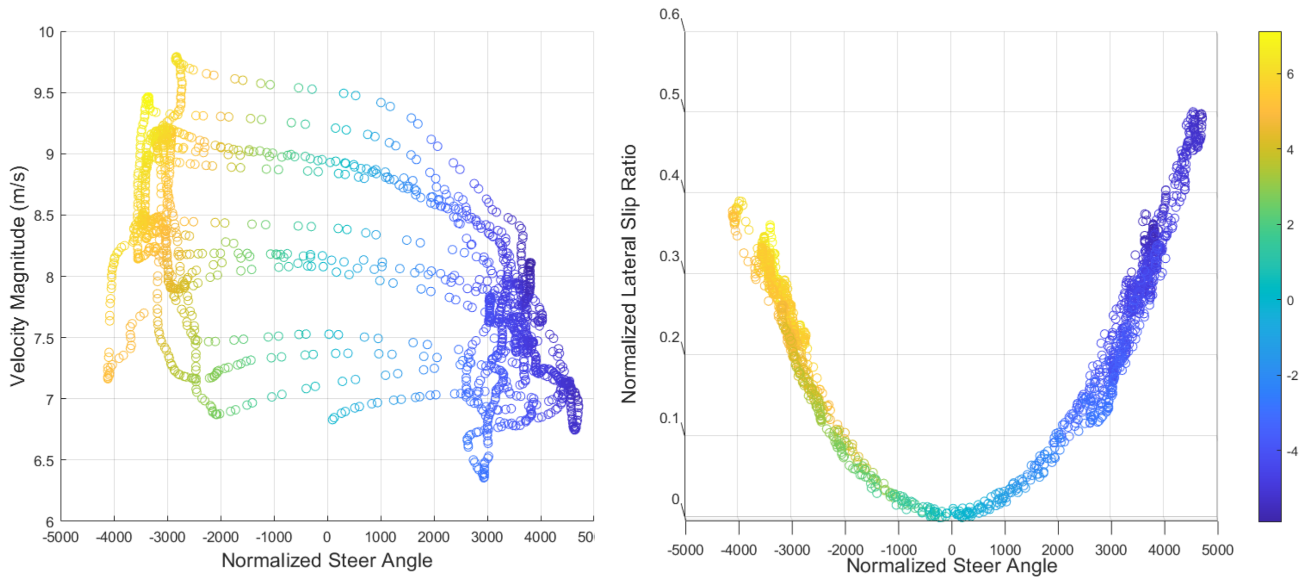


Figure 7.13: The figure-8 training maneuver plotted in the input feature space shaded according to the target lateral acceleration.

The results of the neural network on the test dataset are shown in Figure 7.14. The GRBFNN is able to estimate the target accelerations to within 0.6 m/s^2 in each direction. The error is highest in the lateral direction. This is consistent with the fact that the magnitude of the lateral acceleration is higher than the other components for the figure-8 dataset. The error is roughly evenly distributed, though it is slightly skewed towards the higher dynamic regimes.

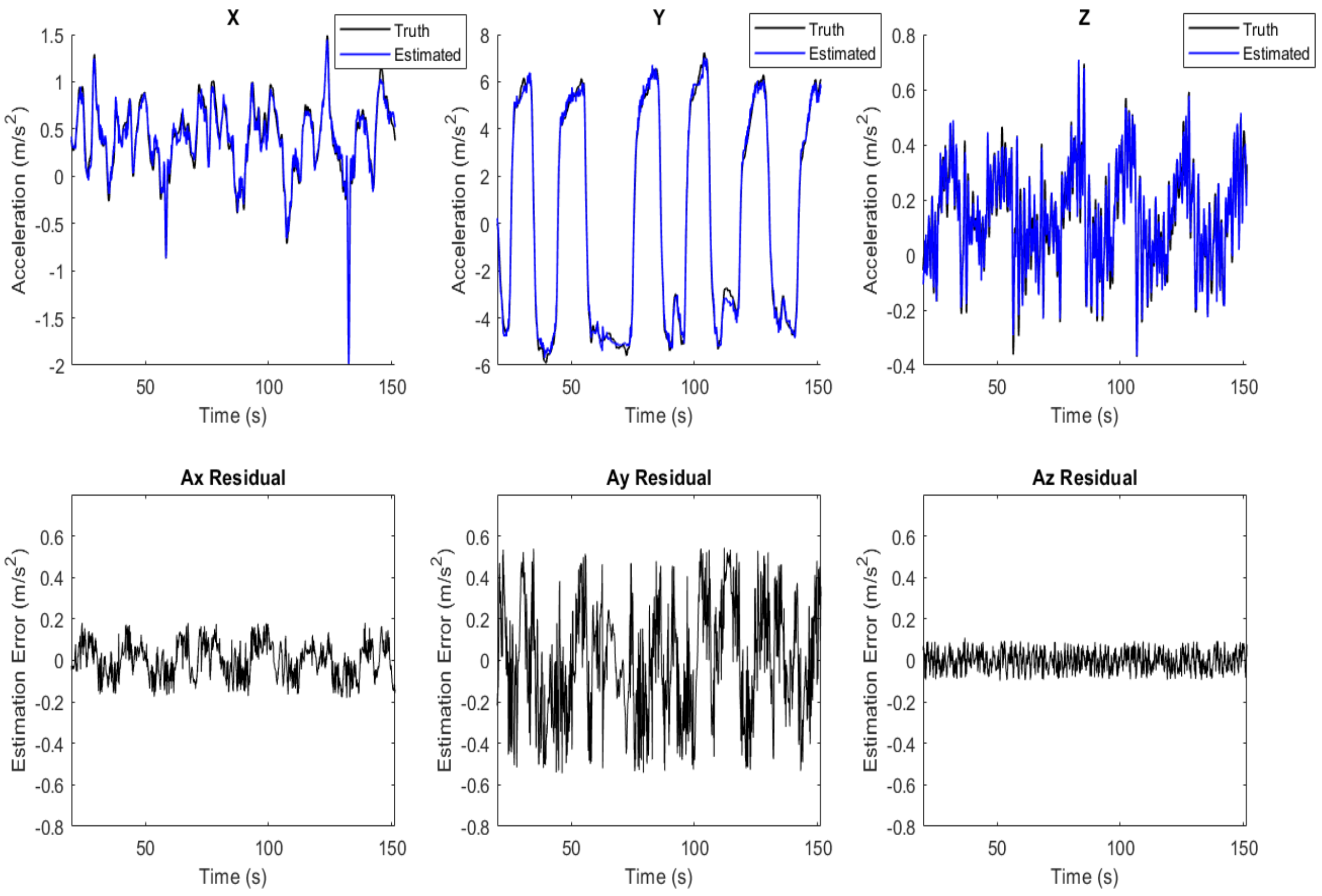


Figure 7.14: Comparison of the true vehicle frame accelerations (Black) to the estimated vehicle frame accelerations (Blue), with the residual estimation error.

Figure 7.15 compares how the training and validation error evolved over the course of training each neural network. In general, the validation error is expected to approach an asymptote at a certain point, and then increase as the neural network overfits the training dataset. The weights at this point are then saved and training is halted. This trend is clearly visible in the x-direction plot. The validation error reaches a minimum around 1800 epochs, and starts to increase afterwards. The training, however, was more unstable in the longitudinal direction than in the lateral direction. The validation error in the lateral decreases steadily towards an asymptote at a mean-squared error of around 0.5. This indicates that the signal-to-noise ratio in the lateral direction is greater than in the longitudinal direction, which is consistent with the greater observability of the lateral dynamics than the longitudinal dynamics for a figure-8 maneuver on flat ground.

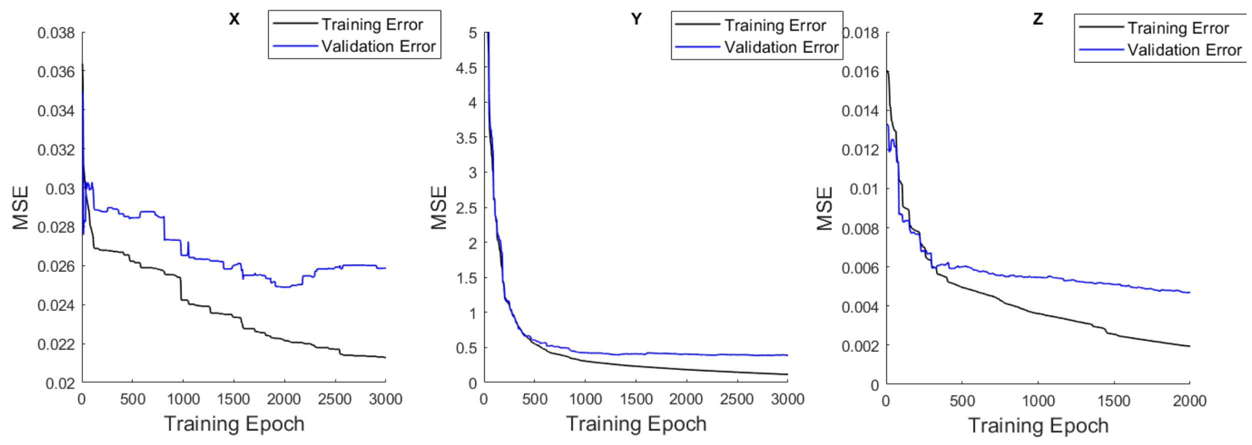


Figure 7.15: Comparison of validation error and training error as a function of the training epoch.

The GRBFNN was able to learn the vehicle dynamics for a figure-8 maneuver to within 0.2 m/s^2 in the longitudinal direction, 0.6 m/s^2 in the lateral direction, and 0.1 m/s^2 in the vertical direction. While the training dynamics were not perfectly stable in every case, each neural network still converged to a state with relatively little error, producing a reliable IMU emulation model. The improved input feature space likely played a role in enabling the neural network to model the vehicle frame accelerations to the observed accuracy.

7.2.3 Pose Estimation

The pose estimation was performed using the covariance-matrix adaptation genetic algorithm described in Section 7.2. However, the estimated parameter vector was updated to include the accelerometer biases of the IMU-to-be-calibrated. This was done to compensate for the non-negligible bias of the commercial-grade Vectornav IMU. The accelerometer biases are therefore estimated in parallel with the 6-D pose of the IMU.

Figure 7.16 compares the measured sensor frame accelerations to the estimated vehicle frame accelerations transformed into the sensor frame. These plots demonstrate how the optimizer estimates the sensor pose. The IMU measures lateral and longitudinal accelerations that contain out-of-phase cyclic components. In the vehicle frame, only the lateral acceleration is cyclical. This indicates that the IMU exists at some yaw relative to the vehicle. In addition, the static accelerations show that the gravity vector in the sensor frame does not point straight down, as it does in the vehicle frame. This further indicates that there are pitch and roll misalignment angles between the IMU and the vehicle frame. The effect of the lever arm is less immediately visible, although it is present in some of the secondary cyclic effects. The measured accelerations are noticeably more noisy than the transformed accelerations. This is due to the lower quality of the Vectornav IMU relative to the ground truth system. Figure 7.17 shows how each individual component of the transformed accelerations compares to the measured quantities. The residual error is relatively evenly distributed, which suggests that the sensor pose estimate is not biased.

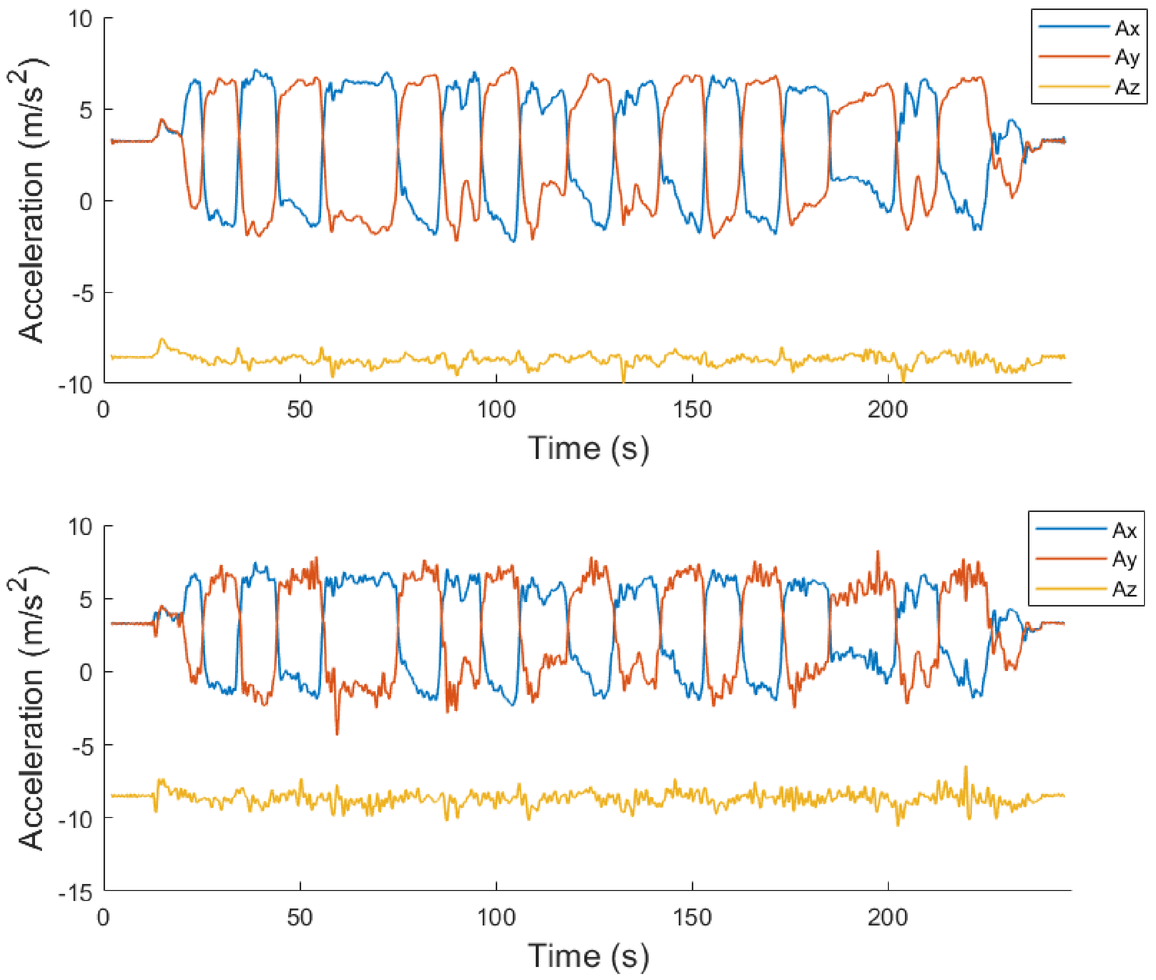


Figure 7.16: All accelerations transformed from the vehicle frame to the sensor frame (above) compared to all accelerations measured in the sensor frame (below).

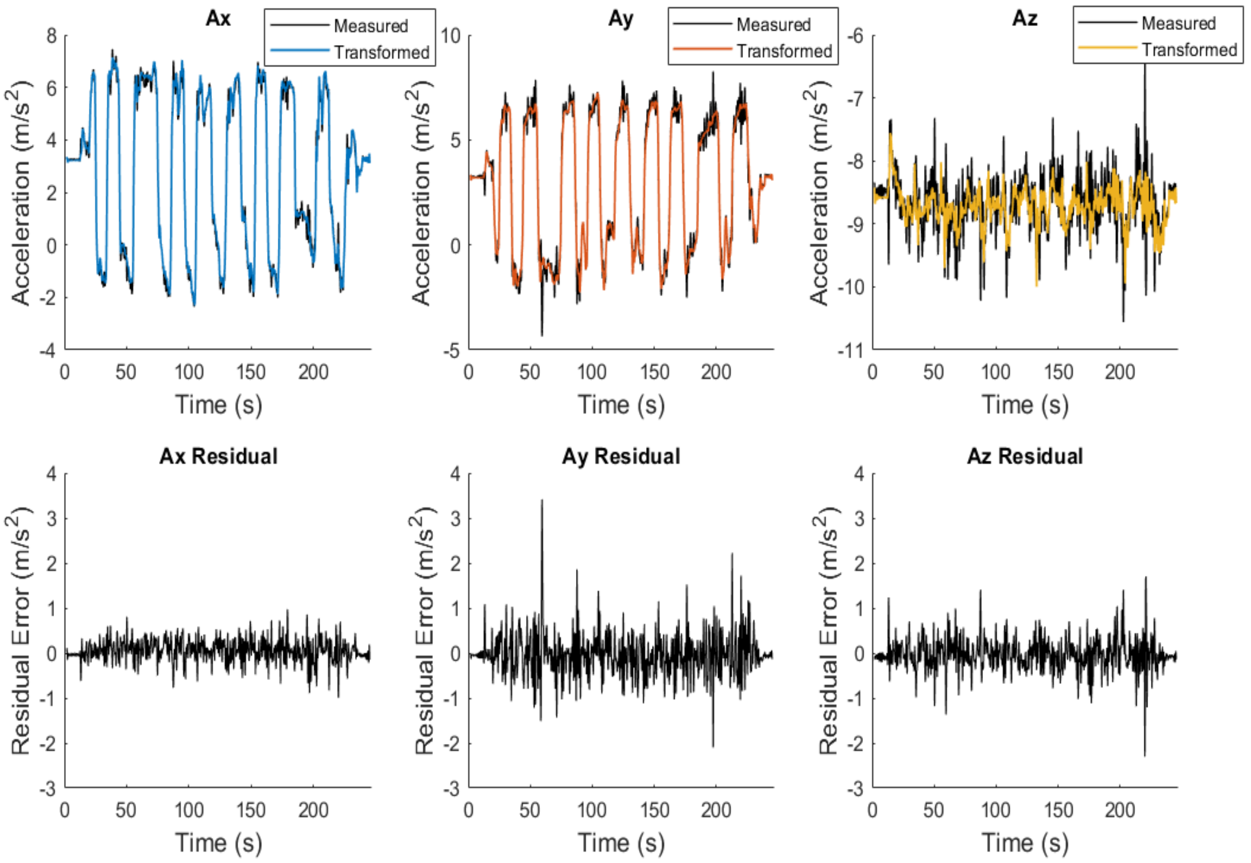


Figure 7.17: Measured sensor frame accelerations compared to estimated accelerations transformed from the vehicle frame (Top), with residual error (Bottom).

The pose estimation error for the figure-8 dataset was on average relatively low, despite the short duration of the test routine. The longitudinal and lateral position errors were 1.3 cm and 4.5 cm respectively. The vertical position error was 25.6 cm, which is considerably worse than the error in the other two directions. However, this result was expected given the much lower observability of the vertical dynamics in this dataset. The orientation error, on the other hand, was unexpectedly high for the figure-8 routine. An error of 3.6 deg in roll, 3.6 deg in pitch, and 2.2 deg in yaw was obtained.

The relatively low position error is likely due to the higher observability of the lever arm under the highly dynamic figure-8 maneuver. The high angular rate under these dynamic regimes induces a greater acceleration at a specified distance from the center of gravity. This increases the observability of the lever arm, which in turn allows the optimizer to more easily estimate its true value.

Sensitivity Test

With the model accuracy demonstrated previously for the figure-8 dataset, a reasonably accurate pose estimate can be obtained. However, it would be ideal if the relationship between the model error and the pose estimation error could be characterized. A sensitivity test was conducted to assess this relationship. IMU models of varying accuracies were approximated by adding Gaussian distributed error to the ground truth accelerations for a range of means and standard deviations. The corrupted signals were then passed into the optimizer, producing a pose estimate. The error in the estimation was then characterized as a function of the mean and standard deviation of the added noise. It was determined that the pose estimation error is correlated far more strongly with the standard deviation of the model error than its mean. Therefore only the results for the standard deviation are presented in this thesis.

Figure 7.18 shows how the error in the lever arm varies with the standard deviation of the model error. The error bars encompass uncertainty due to differences in the mean of the model error, as well as differences from trial to trial. The error increases approximately proportionally with the standard deviation, such that it is only below 5 cm in the longitudinal and lateral directions when the model error is at or below a standard deviation of 0.2 m/s^2. The vertical error is much larger than the longitudinal or lateral errors. This makes sense, given that the observability of the vertical dynamics is low for maneuvers that are conducted on flat ground.

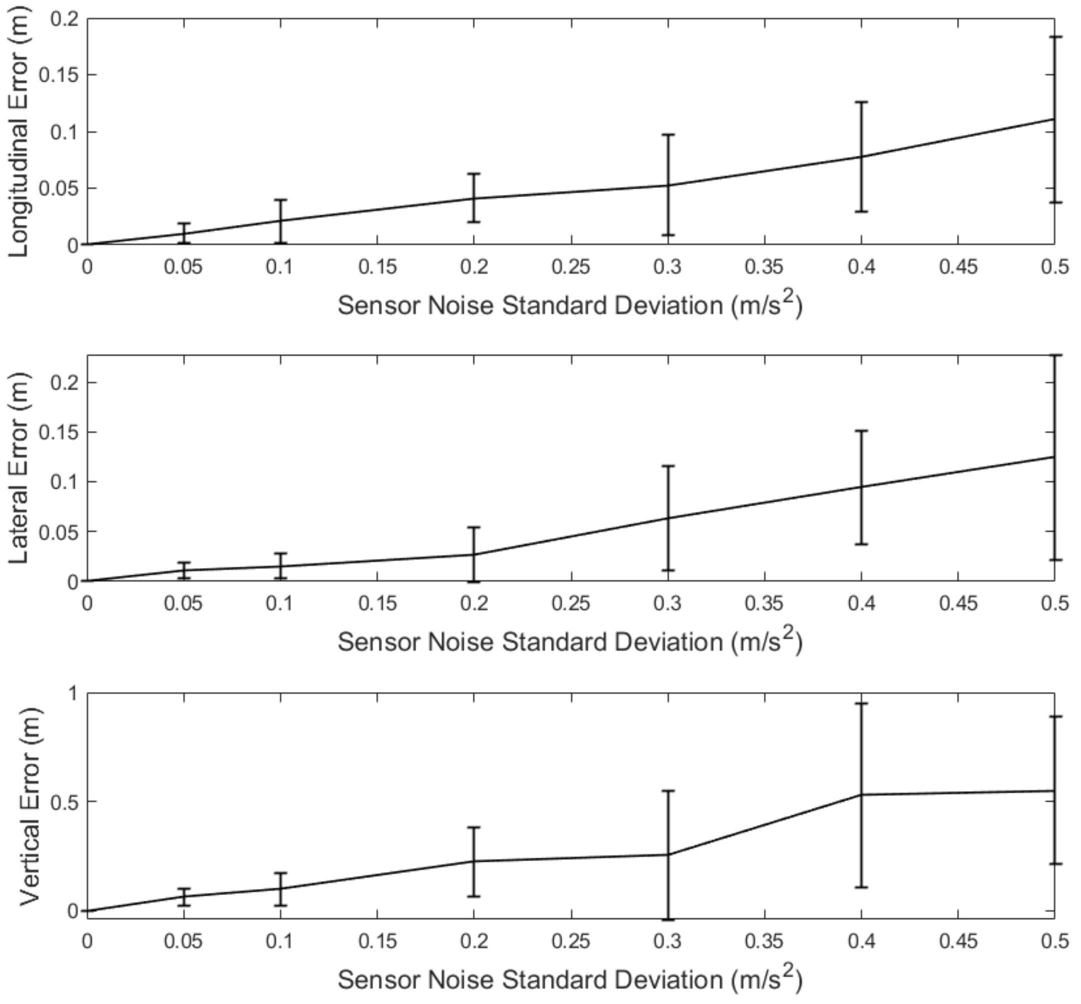


Figure 7.18: Individual sensor position estimates as a function of the standard deviation of the model error.

The relationship between the orientation error and the model error, shown in Figure 7.19, is similar to that of the lever arm. A model error of less than 0.2 m/s^2 corresponds to a an error of less than 1 deg. The yaw angle error is noticeably less sensitive to the model error than the other two, because the yaw dynamics are more observable for a figure-8 maneuver. Even with a model error of 0.5 m/s^2, the average yaw error remains below $1\circ$.

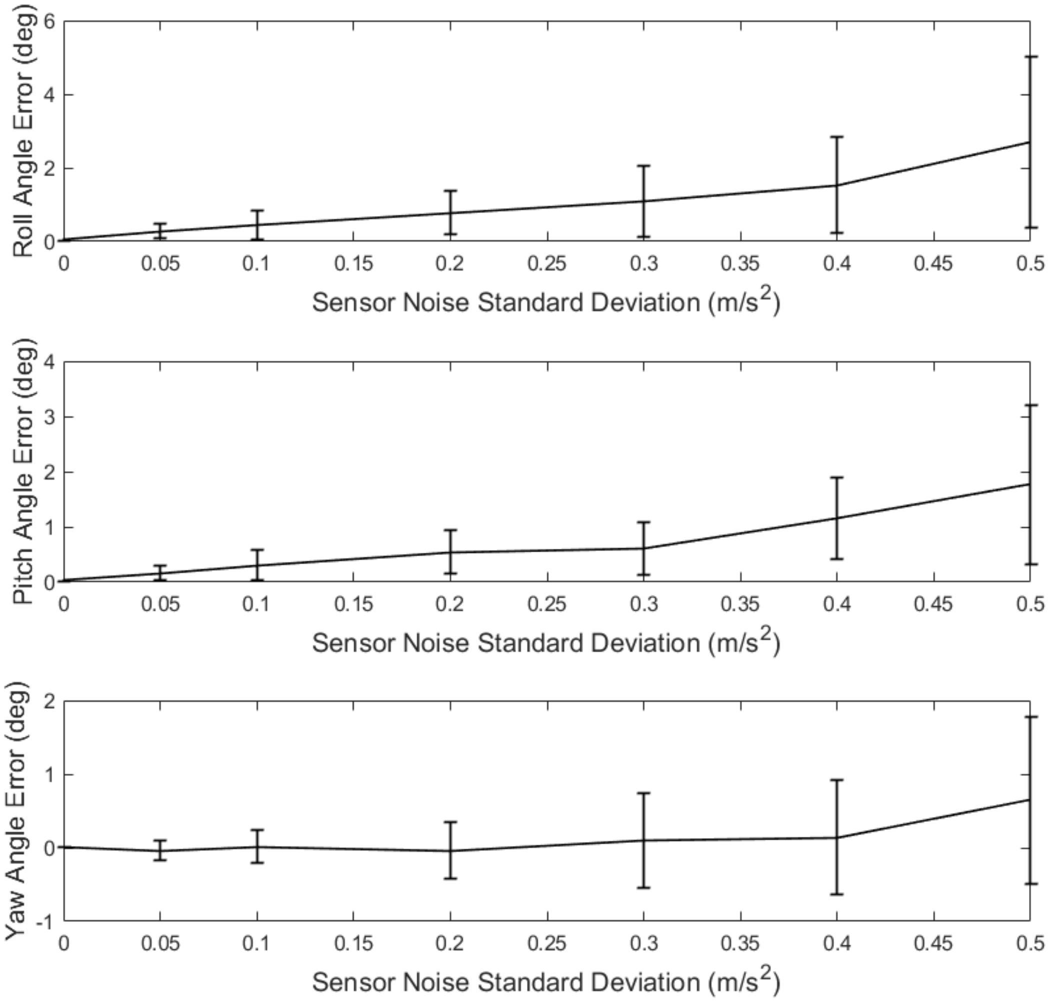


Figure 7.19: Individual sensor orientation estimates as a function of the standard deviation of the model error.

Figure 7.20 shows the relationship between the total position error and the standard deviation of the model error. The total error is dominated by the vertical error, as expected. The average error in the estimated lever arm is 10 cm when the model estimation error is 0.1 m/s^2.

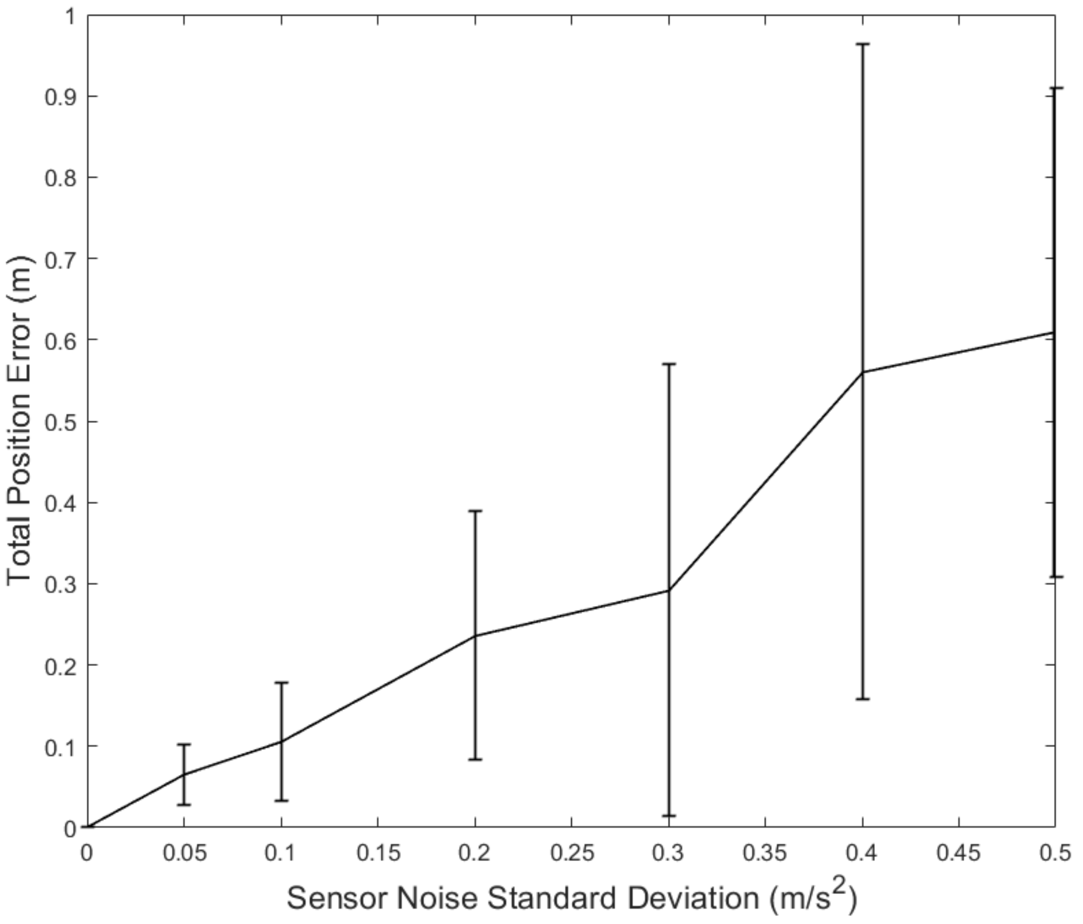


Figure 7.20: Total position error as a function of the standard deviation of the model error.

The total orientation error increases slowly, but non-linearly with the model error, as shown in Figure 7.21. The average only exceeds 1 deg between 0.3 m/s^2 and 0.4 m/s^2. The angular error becomes highly variable once the model error reaches 0.5 m/s^2, however, with some estimates at over 5 deg of total angular error.

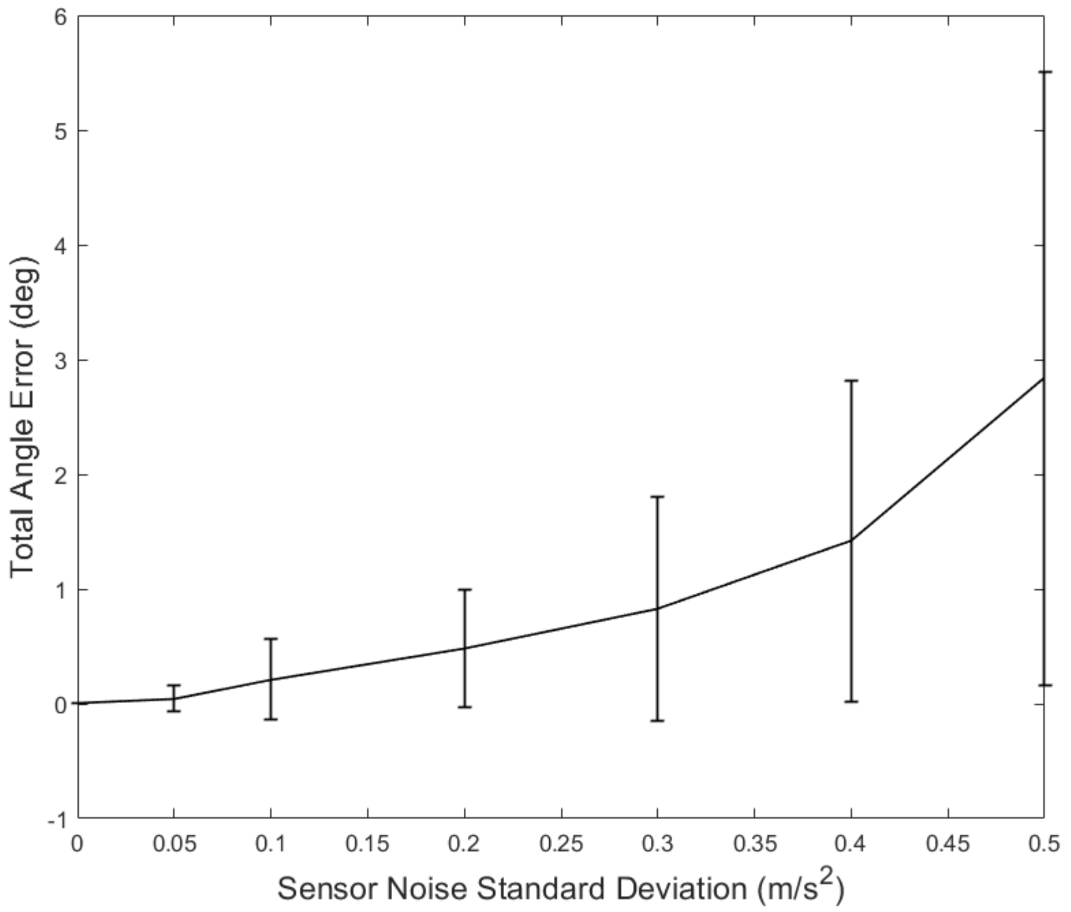


Figure 7.21: Total orientation error as a function of the standard deviation of the model error.

Several conclusions can be drawn from the sensitivity test. The model error needs to be on the order of 0.2 m/s^2 in order to achieve good results for all six elements of the sensor pose. The vertical position error will always be worse than the lateral or longitudinal position errors for a calibration routine conducted on flat ground, such as the figure-8 maneuver. The total position error is dominated by the vertical error, while the yaw angle error is negligible in comparison to the pitch and roll angle errors. In general, the variance in the pose error increases with increasing model error. The test gives a good indication of what level of accuracy is needed in the IMU emulation model in order to achieve the desired pose estimation accuracy. However, this test was run for a fixed time length. It might be expected that the pose error will stabilize at a lower value as the signal accumulates over time. A characterization of the pose estimation error as a function of calibration time is therefore necessary to draw a complete conclusion.

7.3 Square Maneuver Trial

The square maneuver dataset is composed of repeated right angle turns, first to the left, and then to the right. The purpose of this dataset is to test the neural network model's performance on a maneuver that provides some dynamics during turns, but is also a more feasible calibration routine than the figure-8. While a figure-8 maneuver requires a large open space, the square maneuver replicates right angle turns that can be performed wherever there is an intersection on a city grid. Learning the dynamics of this maneuver is therefore of particular importance to urban applications.

7.3.1 IMU Emulation

The square maneuver dataset is similar to the figure-8 dataset in that it is composed of repeated turns that excite the lateral dynamics of the vehicle. However, the square maneuver is composed of blocks of turns, rather than alternating left and right turns. Consequently each turn has a lower maximum steer angle, a lower maximum velocity, and a shorter duration. The vertical dynamics are similarly weak because the maneuver takes place on flat ground. Figure **??** shows the raw input measurements for the square maneuver dataset.

Aside from differences in the order of turns, and the lower velocity during those turns, the square maneuver causes a higher lateral slip than the figure-8. Some turns in the dataset induced slip over 1%. Figure 7.22 shows the normalized inputs, including the lateral slip percentage. The target accelerations shown in Figure 7.23 are greatest in the lateral direction, and smallest in the vertical direction, as expected. The brake and throttle play a greater role in the longitudinal dynamics for the square maneuver dataset.

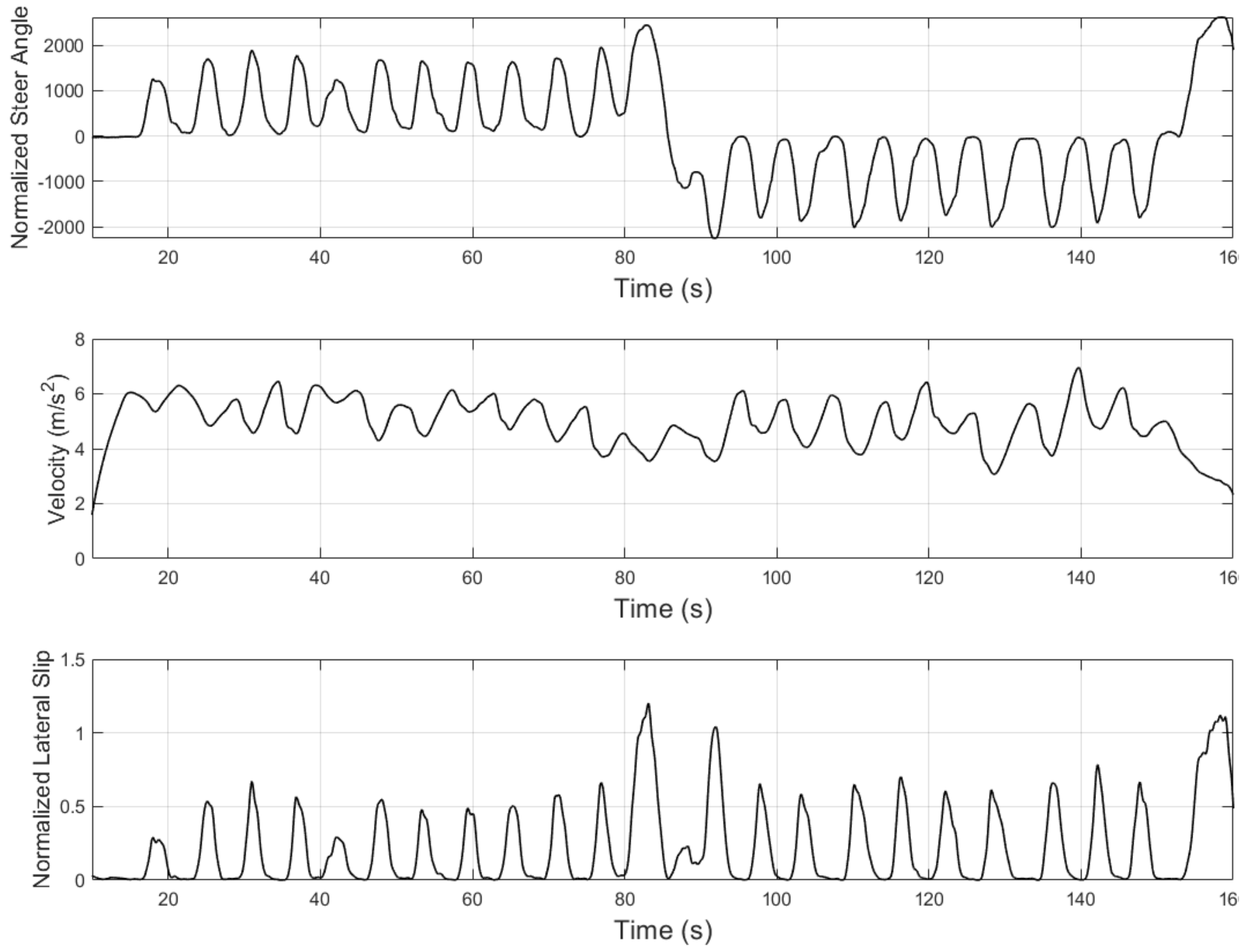


Figure 7.22: Normalized input time series for the Square maneuver dataset.

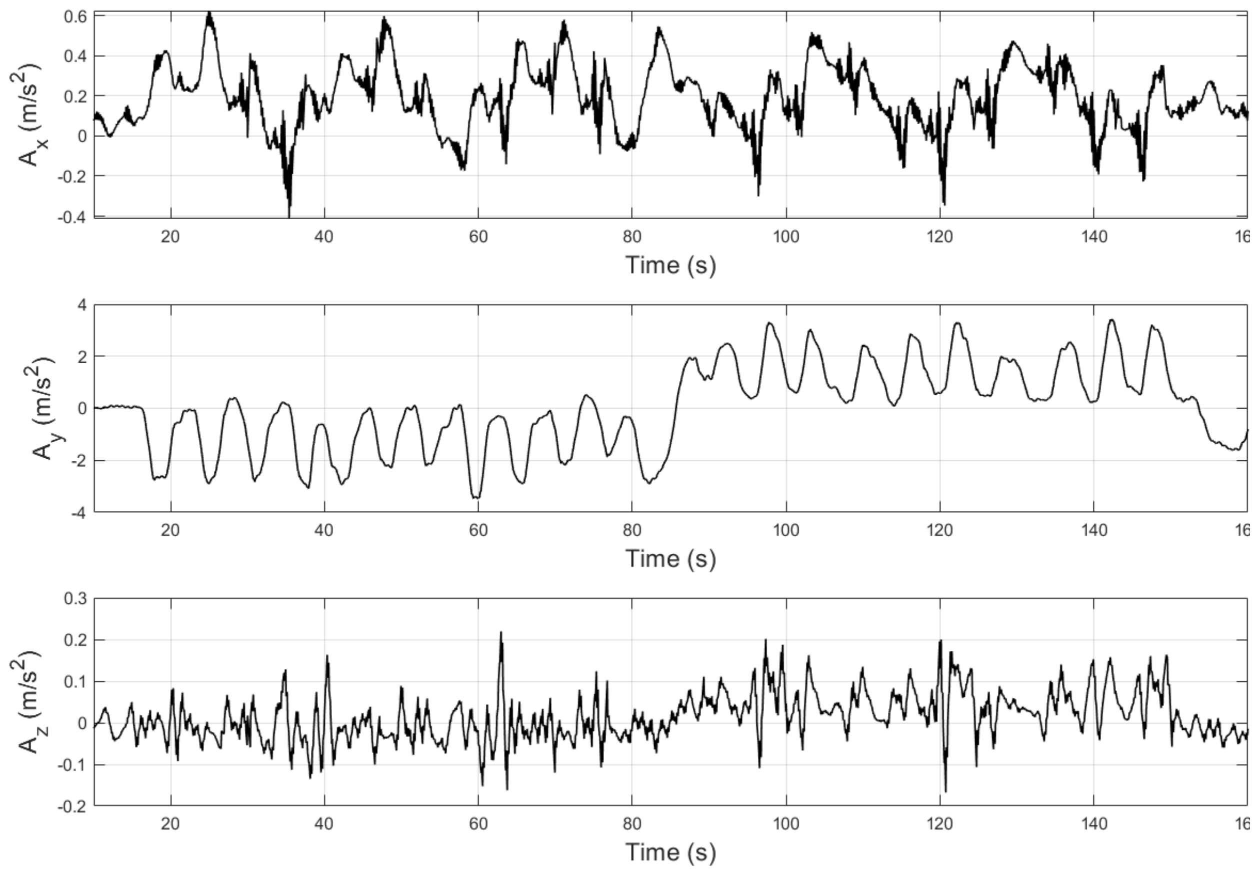


Figure 7.23: Target accelerations for the Square maneuver dataset.

Figure 7.24 shows how the lateral acceleration varies within the input feature space. The square maneuver forms a fan within the input space. The right-hand plot shows the same quadratic relationship between steer angle and lateral slip holds for the square maneuver. This makes intuitive sense given that it is theoretically dependent on the friction properties of the road surface, rather than the maneuver itself.

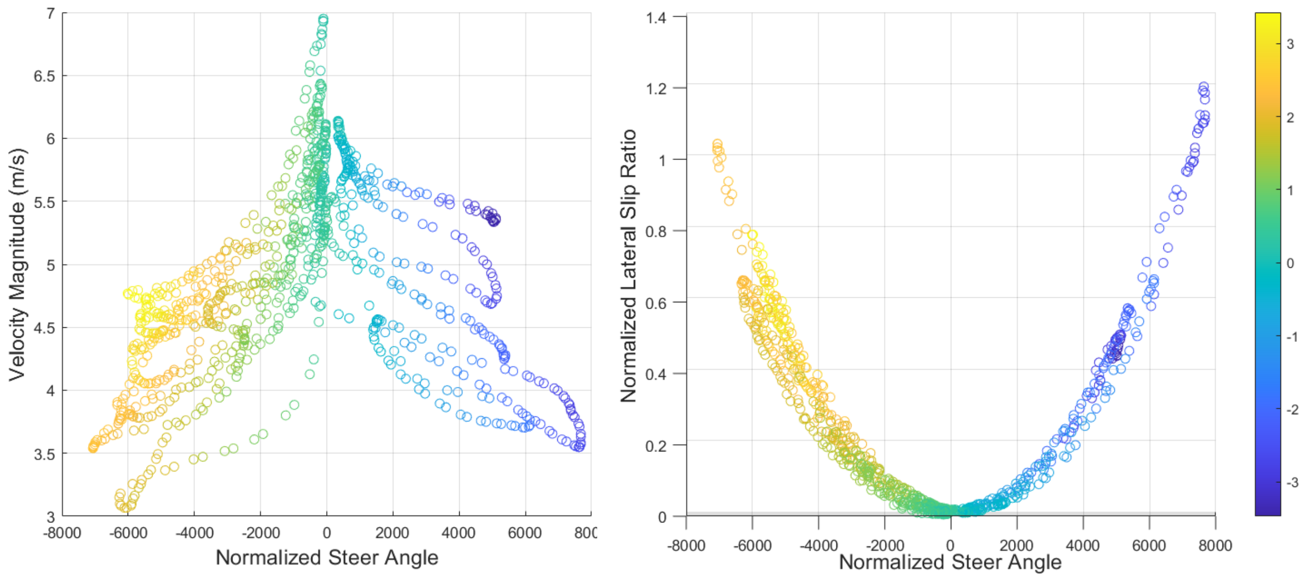


Figure 7.24: The square maneuver dataset trajectory in the input feature space shaded according to the target lateral acceleration.

The model is even more accurate for the square maneuver dataset than it is for the figure-8 dataset. The residual in the lateral direction does not exceed 0.5 m/s^2 in this case. However, there is a marked cyclical component in the residual that might have some effect on the pose estimation process. Figure 7.25 compares the accelerations estimated by the model to truth. The training dynamics, shown in **??**, are ideal. The validation error either bottoms out before reaching an asymptote, or it steadily descends towards that asymptote from the beginning of the training process.

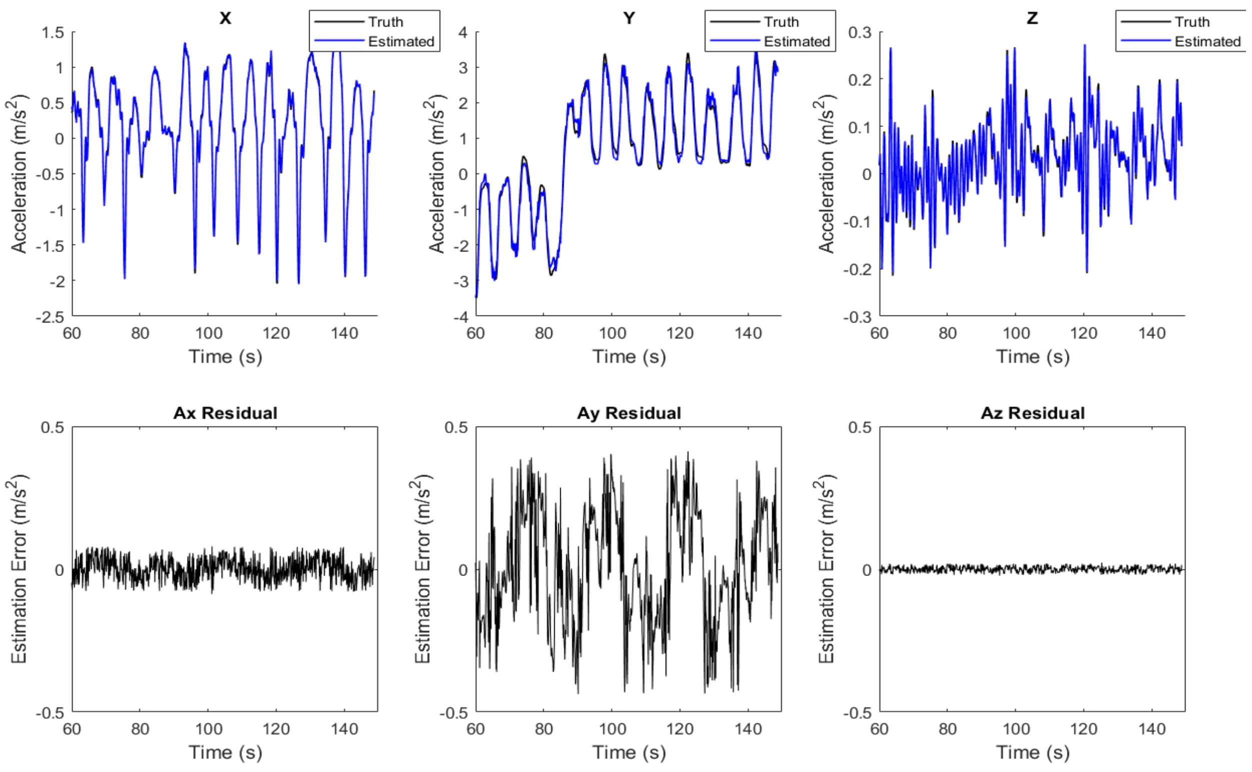


Figure 7.25: Comparison of the true vehicle frame accelerations (Black) to the estimated vehicle frame accelerations (Blue), with the residual estimation error.

The square dataset is an important test for the calibration routine, because it is composed of maneuvers that can, and likely would, be performed by end users. Of the maneuvers that are commonly performed when driving in an urban environment, the right angle turn generates the highest dynamics. The strong performance of the neural network model on this dataset is an encouraging sign.

7.3.2 Pose Estimation

The square maneuver is similar to the figure-8 maneuver in most respects. It was therefore anticipated that the performance of the calibration algorithm would remain approximately the same for the square maneuver dataset. However, the right angle turns in the square maneuver do not excite the lateral dynamics as much as the figure-8 maneuver, which suggests that the lever arm would be less observable for this dataset.

Figure 7.26 shows how the transformed accelerations compare to the measured sensor frame accelerations for the square dataset. The transformed accelerations match the measured accelerations very well, except for some higher frequency secondary effects around the peaks and troughs of the measured accelerations. The measured vertical acceleration also appears to be more variable than the transformed accelerations. Whether this is due to unfiltered noise, or some real dynamics, is unknown.

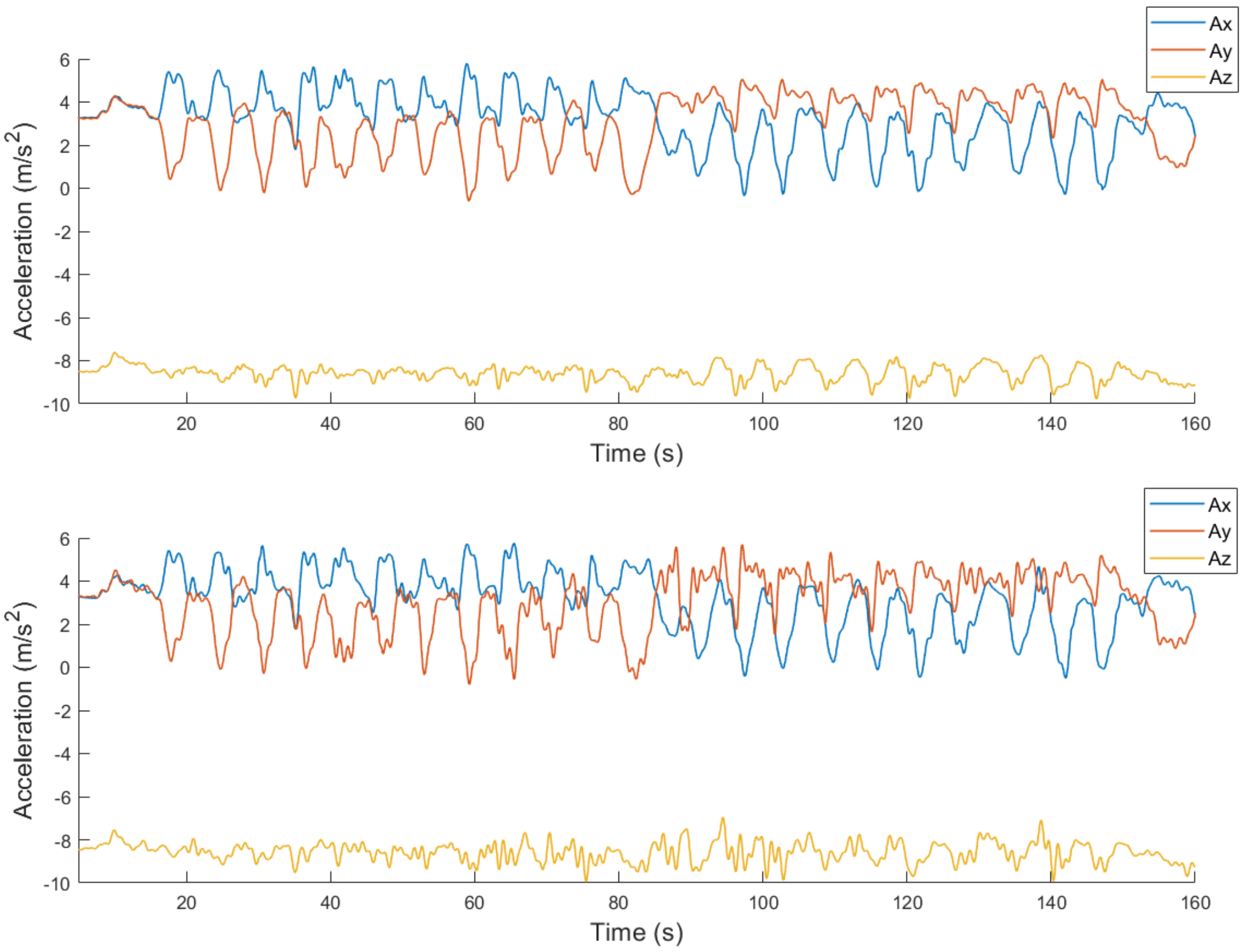


Figure 7.26: All accelerations transformed from the vehicle frame to the sensor frame (above) compared to all accelerations measured in the sensor frame (below).

The residual error shown in Figure 7.27 is approximately the same as for the figure-8 dataset. However, there is an increase in the residual error around the middle of the trial, and in particular around the transition between the clockwise and counter-clockwise portions of the dataset. This effect could be a result of higher model error in the transition period.

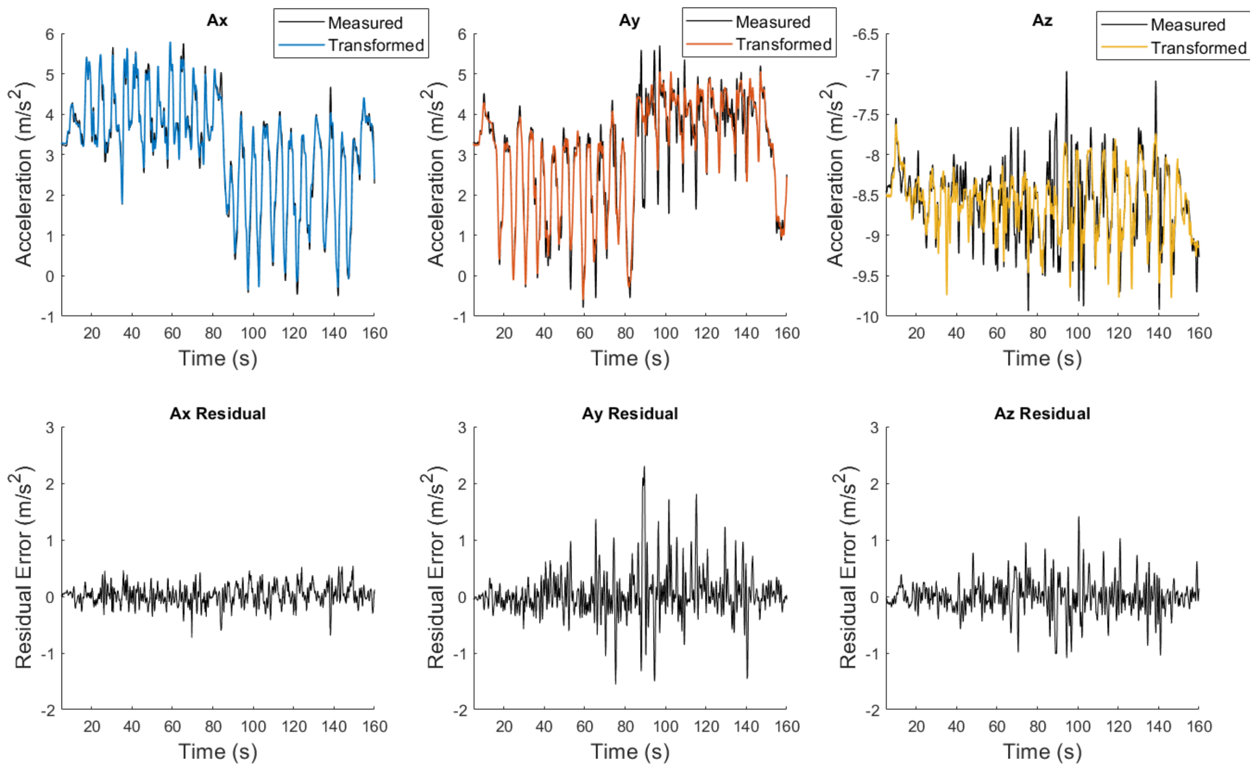


Figure 7.27: Measured sensor frame accelerations compared to estimated accelerations transformed from the vehicle frame, with residual error.

The error in the orientation estimate for the square maneuver dataset was lower than that of the figure-8 dataset in each direction. An error of 0.32 deg in roll, 0.34 deg in pitch, and 0.56 deg in yaw was obtained for the test routine. The position error, meanwhile was comparable in the lateral direction, but much worse in the longitudinal and vertical directions. The IMU position estimation error was 30.4 cm in the longitudinal direction, 2.22 cm in the lateral direction, and 69.6 cm in the vertical direction.

The square maneuver could potentially be a viable calibration maneuver in practice, especially in urban environments, where intersections are common. If it is paired with a maneuver that adequately excites the longitudinal and vertical dynamics of the vehicle, the combination could be recommended as a standard calibration routine. This would ensure that the vehicle can be calibrated properly in the expected individual use case.

7.4 Road Drive Trial

The road drive dataset was collected along a section of paved road in rural Alabama. The chosen road has a number of sharp turns that excite the vehicle's lateral dynamics, as well as a number of positive and negative vertical slopes. It was anticipated that these slopes would excite the vertical dynamics, increasing the observability of the vertical component of the lever arm. The road drive trial simulates the most flexible calibration routine possible. No special maneuver is required from the driver aside from what they would do on a daily basis.

7.4.1 IMU Emulation

Figure **??** displays the raw input time series for the road drive dataset. Along straight sections of the road, the steer angle is close to zero, and consequently the yaw rate and lateral velocity are close to zero as well. However, there are a few spikes that correspond to sharp bends in the road, or turns made to get on or off the highway. The average velocity is substantially higher than for the figure-8 or square maneuver datasets, because the vehicle was driven at the particular speed limit of the road. The increased velocity amplified the dynamic response of the vehicle, resulting in greater lateral accelerations even for relatively small steering angles.

The normalized inputs fed to the neural network model are presented in Figure 7.28. The lateral slip is negligible except when the vehicle is performing a turn. These turns were executed at low velocities, which limited the dynamics that could be obtained from them. However, the target time series show enough variation to assume that the dynamics were excited to a sufficient degree to make the IMU's lever arm observable.

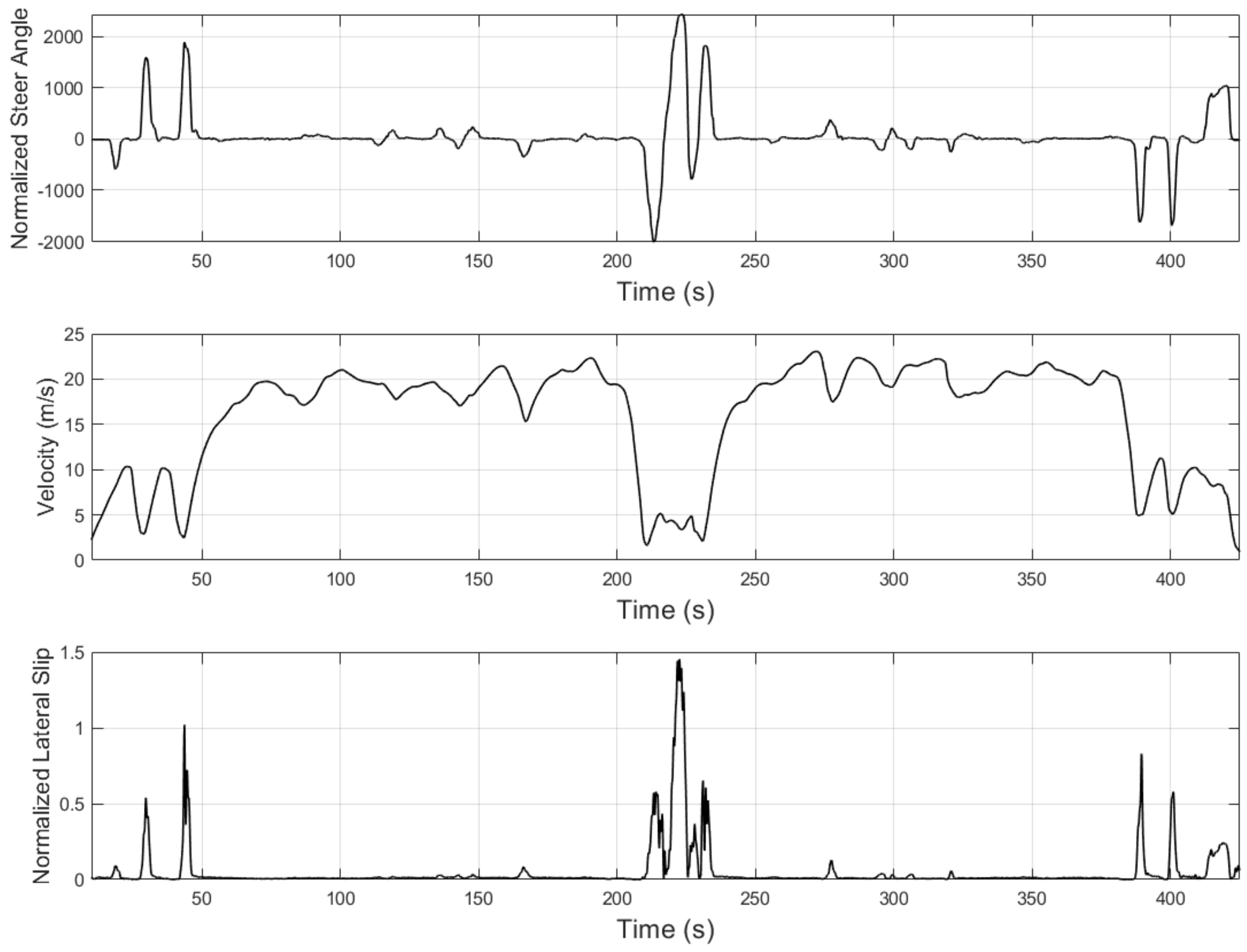


Figure 7.28: Normalized input time series for the Road drive dataset.

Figure 7.29 shows how the target accelerations varied across the road drive. A number of spikes of similar magnitude are present in both the longitudinal and lateral dynamics. Some indicate right angle turns at low speed, while others indicate slight turns taken at the speed limit. While the maximum lateral acceleration for the figure-8 dataset was around 8 m/s^2, the magnitude of the lateral acceleration for the road drive only reaches 6 m/s^2 at most. However, the vertical acceleration does reach a greater maximum value than it does for the figure-8 or square maneuver datasets. This is likely due to non-zero grade in the road, which causes the gravity vector to shift to and away from true vertical.

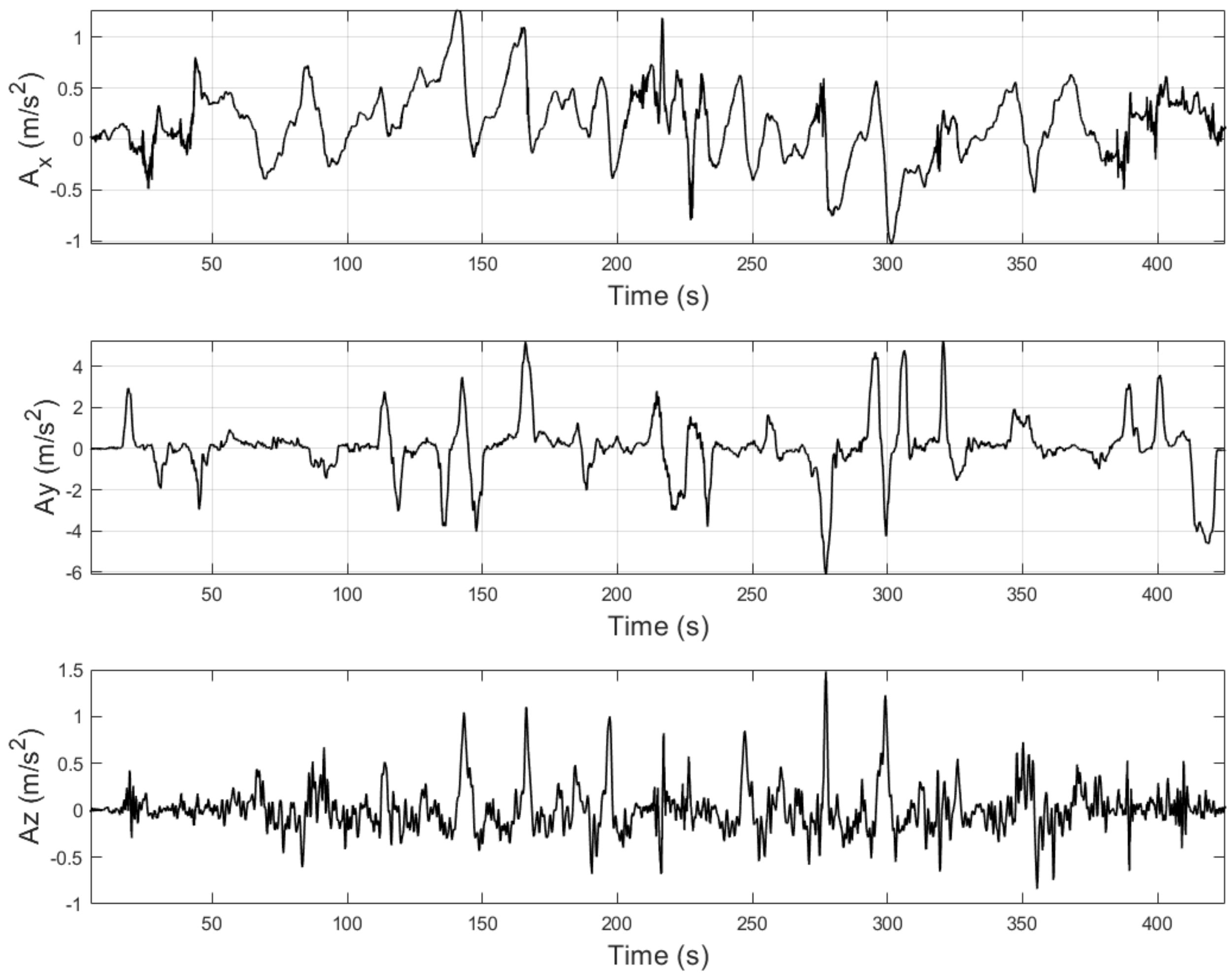


Figure 7.29: Target accelerations for the Road drive dataset.

The trajectory of the road drive through the input feature space is shown in Figure 7.30, shaded according to the lateral acceleration. The steering angle varies widely at low velocities, but is much tighter at higher velocity. Despite this, the lateral acceleration varies more strongly at higher velocity. The amplified dynamics at high velocity increase the observability of the lever arm.

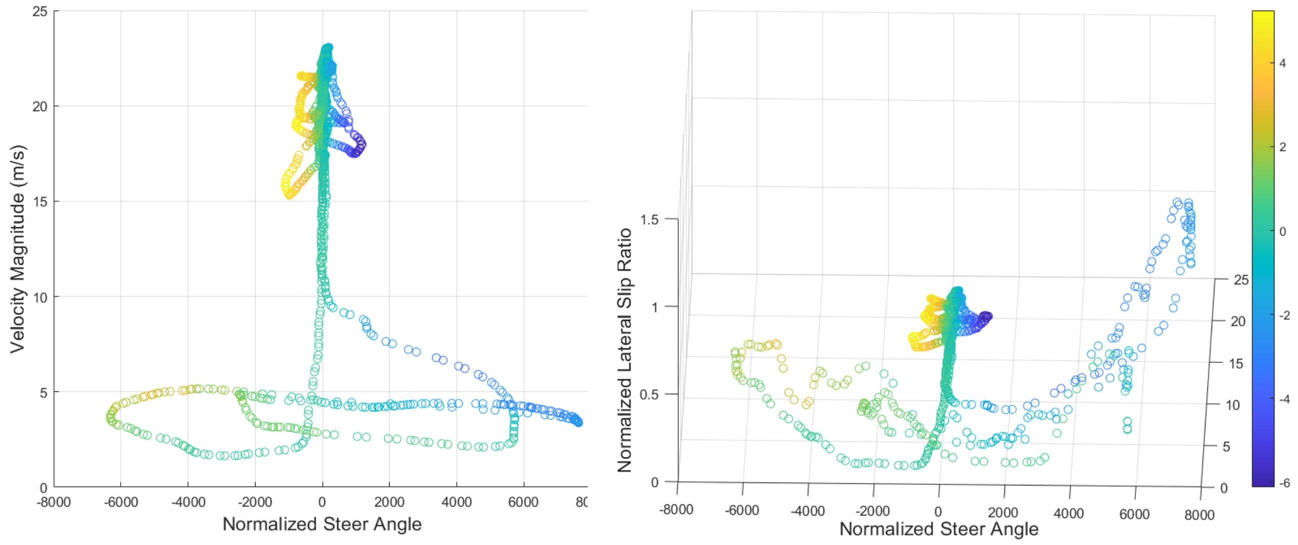


Figure 7.30: The road drive dataset trajectory in the input feature space shaded according to the target lateral acceleration.

Figure 7.31 compares the accelerations predicted by the model with the ground truth accelerations measured in the vehicle frame. The model is less consistent for the road drive dataset than it is for the figure-8 or square maneuver datasets. While it generally tracks within 1 m/s^2 of the ground truth, the model performs noticeably worse over certain intervals of the test dataset. These intervals generally correspond to regions of low velocity and high lateral slip. This indicates that the model was not trained on a sufficiently comprehensive dataset. This issue could easily be corrected by pairing the road drive with another training dataset composed of maneuvers at low velocity and high slip, such as the figure-8 or square maneuvers.

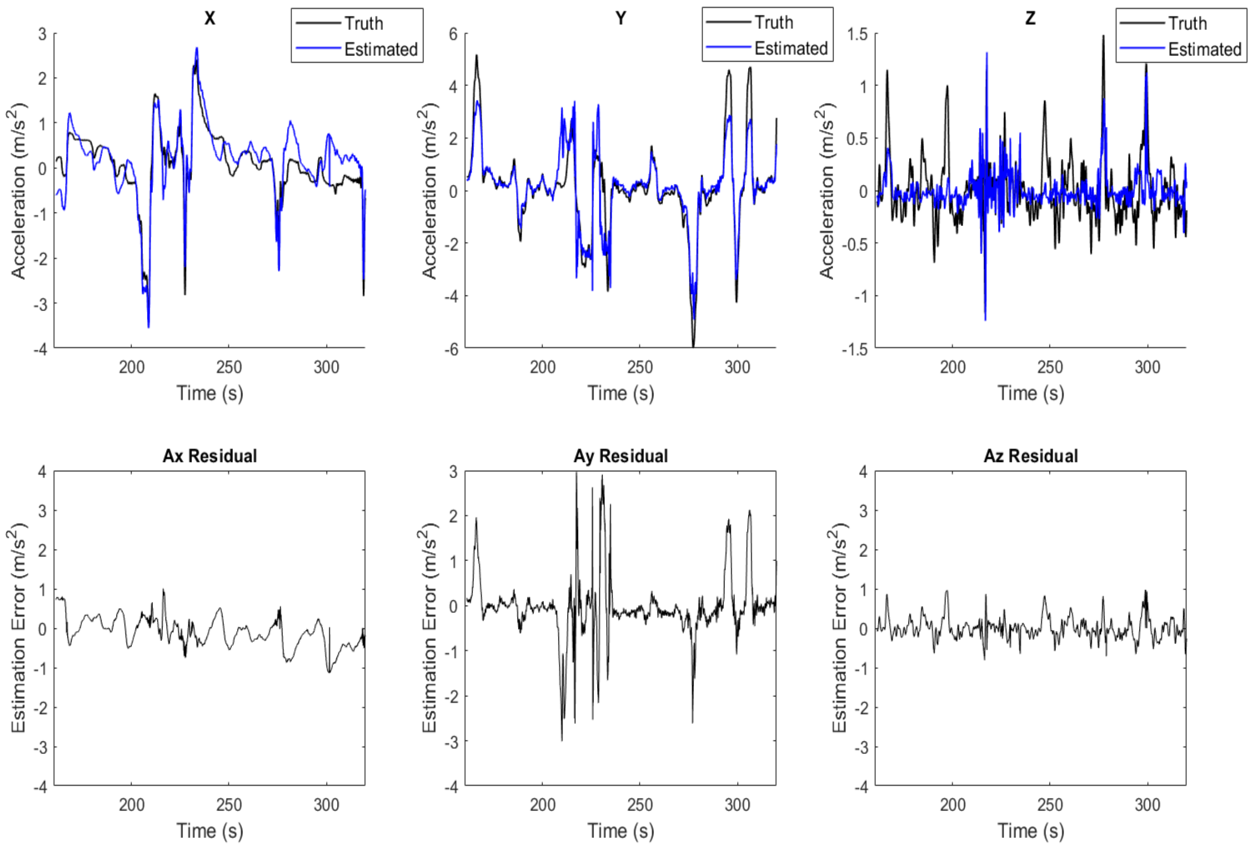


Figure 7.31: Comparison of the true vehicle frame accelerations (Black) to the estimated vehicle frame accelerations (Blue), with the residual estimation error.

The training dynamics are correspondingly less stable for the road drive than for the figure-8 or square maneuver datasets. Figure **??** shows how the validation error in the longitudinal and vertical directions increases with the number of training epochs, rather than decreasing. This is the opposite of the expected pattern, indicating that the neural network was not able to learn a general formula for the dynamics. The validation error in the lateral direction, in contrast, displays the expected behavior.

The results of the IMU emulation model on the road drive dataset show that it can be applied successfully, not only on controlled, well-defined maneuvers, but also on a freely chosen itinerary. The road drive trajectory through the input feature space was consequently less dense than the figure-8 or square maneuver datasets. Despite this, the GRBFNN was able to model the vehicle accelerations when trained only on data from the road drive. However, the model did perform worse at some intervals than the standard set by the other two datasets. This result was expected from a maneuver that varies widely in the input space, and could be addressed in the future by a more comprehensive training routine.

7.4.2 Pose Estimation

The measured accelerations in the sensor frame for the road drive are compared to the accelerations transformed from the vehicle frame in Figure 7.32. The transformed accelerations align relatively well with the measured accelerations. However, the measured accelerations display a greater degree of noise than for the previously discussed datasets, which could potentially affect the accuracy of the sensor pose estimate.

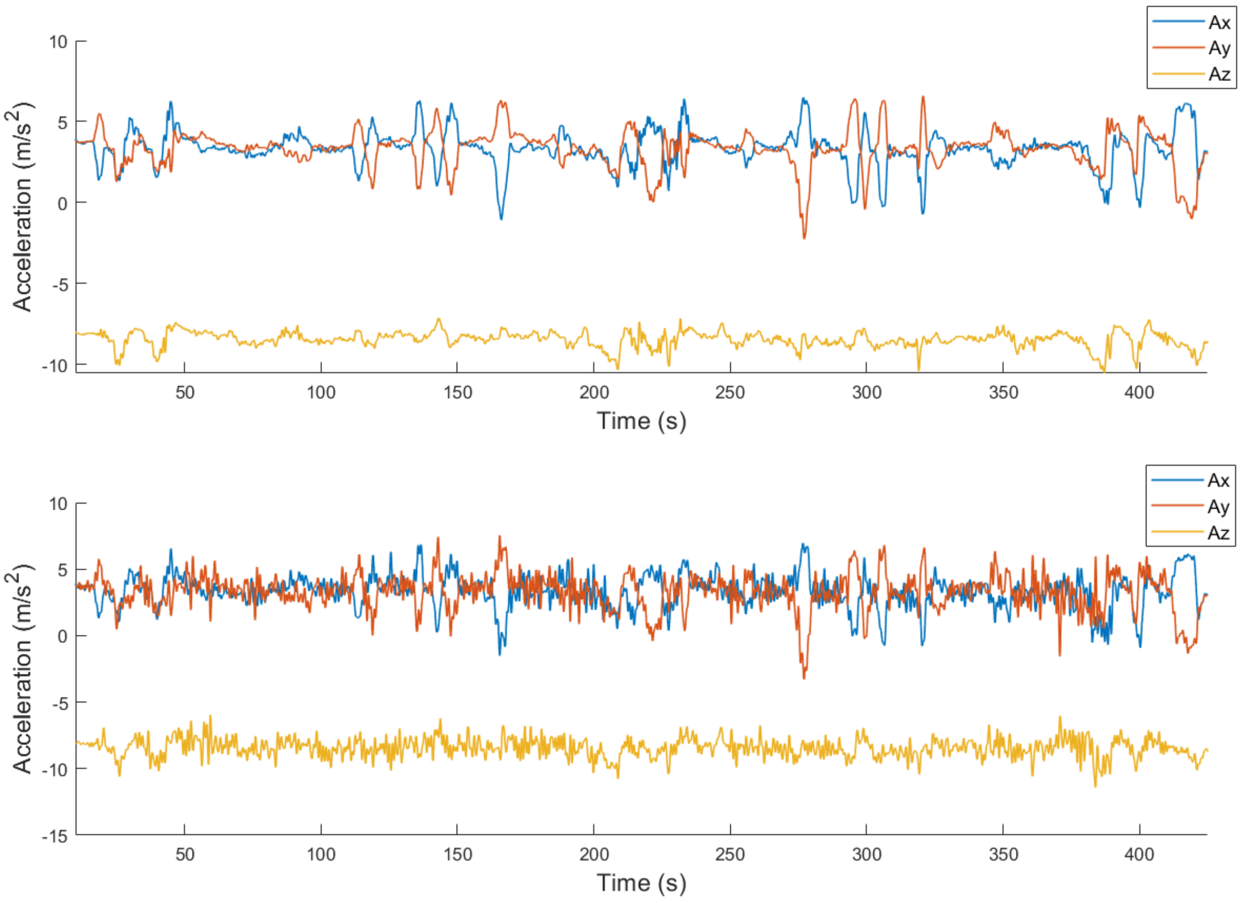


Figure 7.32: All accelerations transformed from the vehicle frame to the sensor frame (above) compared to all accelerations measured in the sensor frame (below).

The residual error between the transformed and measured accelerations, shown in Figure 7.33, is greater in magnitude than for the figure-8 or square maneuver datasets. This may be the result of the additional noise present in the measured accelerations. However, it is likely also the result of the decreased performance of the model on the road drive.

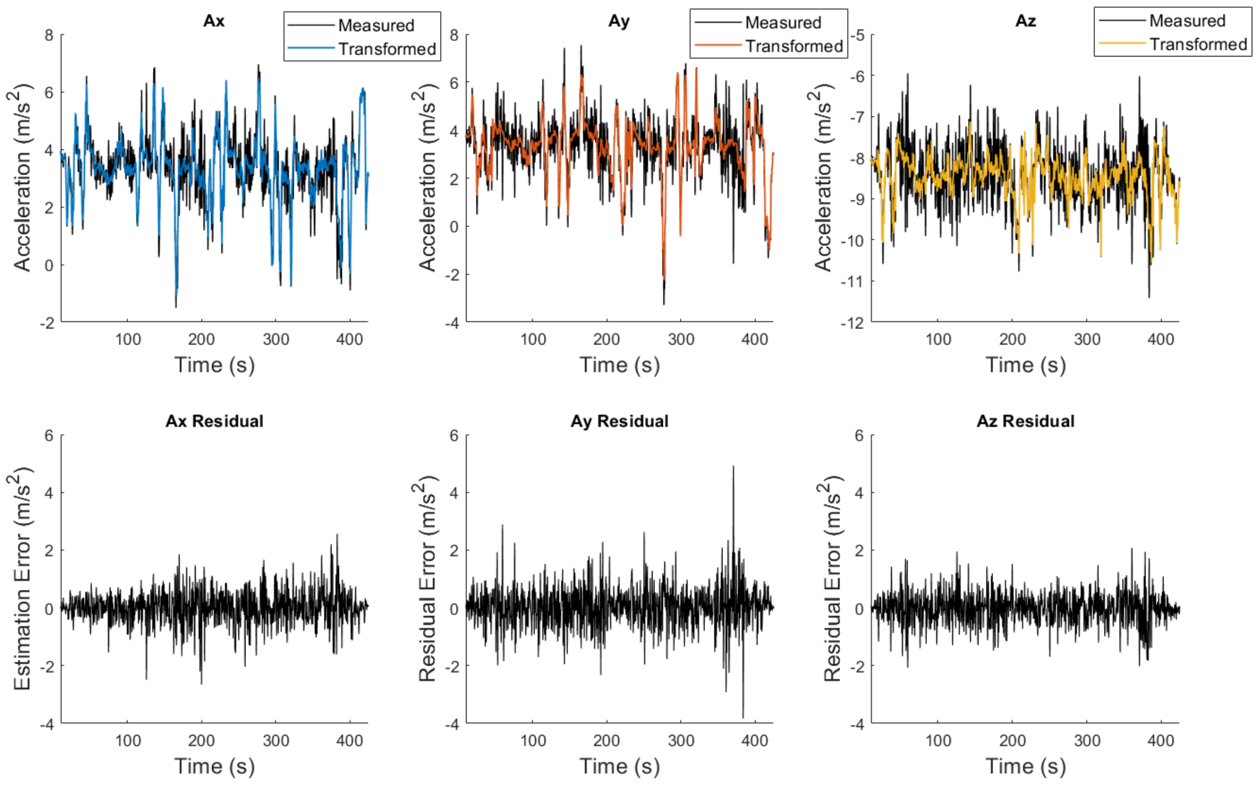


Figure 7.33: Measured sensor frame accelerations compared to estimated accelerations transformed from the vehicle frame, with residual error.

The estimated IMU position error obtained for the road drive dataset was 1.12 m in the longitudinal direction, 37.8 cm in the lateral direction, and 91.1 cm in the vertical direction, which coincides with the expectations outlined previously. The effectiveness of the calibration algorithm is diminished when the model is both trained and tested on data collected on a surface road, rather than in a controlled environment. This could potentially be resolved by training the IMU emulation model on a comprehensive range of controlled maneuvers that spans the input space. The road grade may also have shifted the gravity vector, which would perturb the measured accelerations, disrupting the pose estimation process. This issue is harder to resolve, but could be addressed in the future by a roll and pitch model that corrects for the shift in the gravity vector.

The estimated IMU orientation error, however, was not diminished in comparison to the other two datasets. An error of 1.0 deg in roll, 0.38 deg in pitch, and 0.53 deg in yaw was obtained for the road drive dataset. This result indicates that a surface road drive could be used to effectively estimate the Sensor-to-Vehicle orientation. However, if this routine is to be used for a full extrinsic calibration, it should be combined with another routine for which the lever arm can be more accurately estimated.

7.5 Calibration Trial

The final test of the calibration algorithm is a full calibration trial. The calibration trial tests the performance of the algorithm on a combined routine composed of a static phase, a set of figure-8 maneuvers, and a set of square maneuvers. This routine tests the ability of the neural network model to estimate accelerations from multiple different maneuvers. It is also the ideal calibration routine, because it is composed of maneuvers that maximize the observability of the sensor lever arm. The static phase is important to the calibration routine because it enables the gravity vector to be easily extracted from the measured accelerations. The gravity vector can then be used to produce an initial estimate of the sensor orientation, assuming that the initialization takes place on flat ground. The static phase also stabilizes the pose estimation process, leading to a more accurate final estimate.

7.5.1 IMU Emulation

The IMU emulation model was trained on the figure-8 and square maneuver datasets described previously. It was then tested on the calibration routine. The raw inputs for the calibration routine are shown in Figure **??**. The figure-8 maneuvers were performed first, and possess a higher forward velocity than the subsequent square maneuvers. The normalized input time series are displayed in Figure 7.34. The combination of figure-8 and square maneuvers means that the trajectory through the input space features a variety of steer angles, velocities, and lateral slip percentages normalized by the same scale factor.

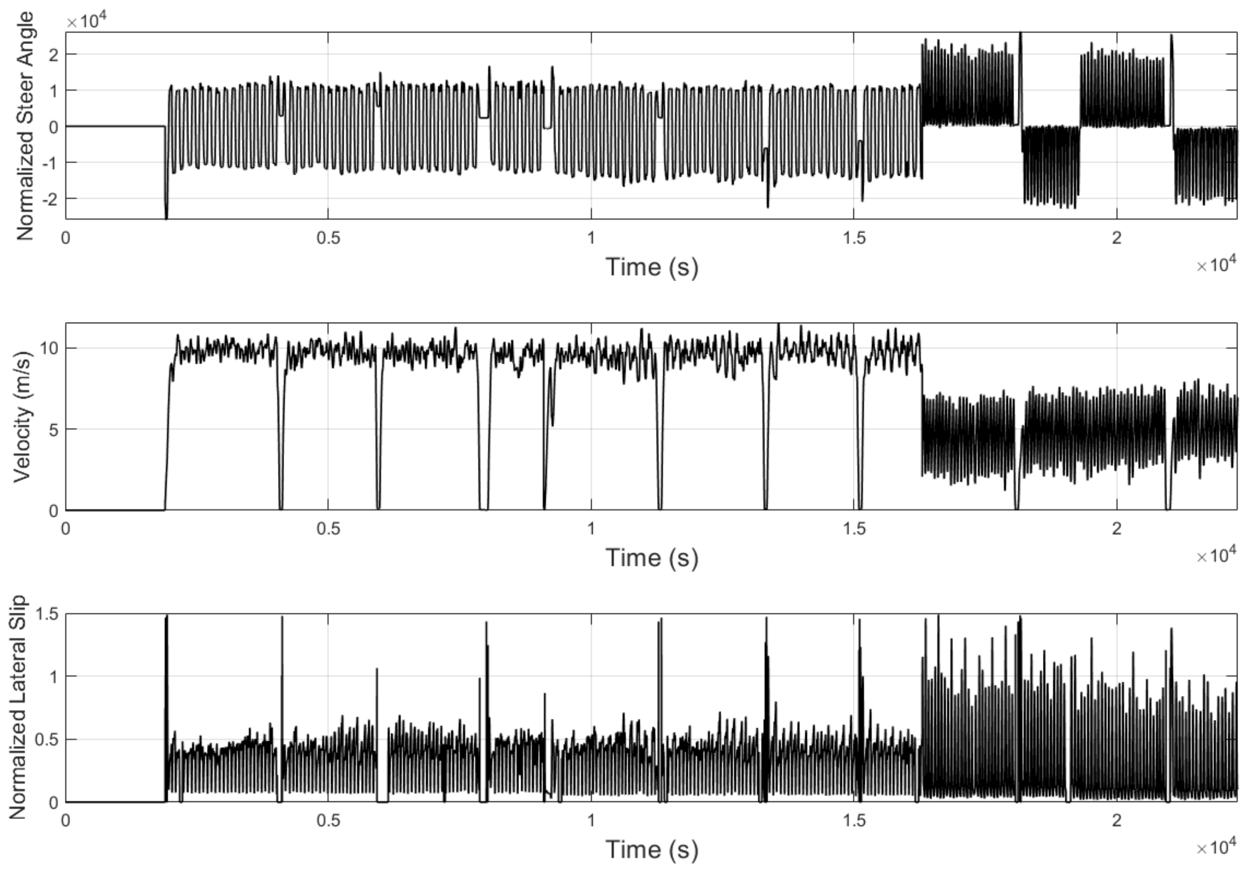


Figure 7.34: Normalized input time series for the Calibration trial dataset.

There is a corresponding degree of variation in the target accelerations, shown in Figure 7.35. The magnitude of the lateral and vertical acceleration is greater for the figure-8 portion of the routine, while the magnitude of the longitudinal acceleration is greater for the square maneuver portion of the routine. The combination therefore maximizes the dynamics in all three directions within one test dataset.

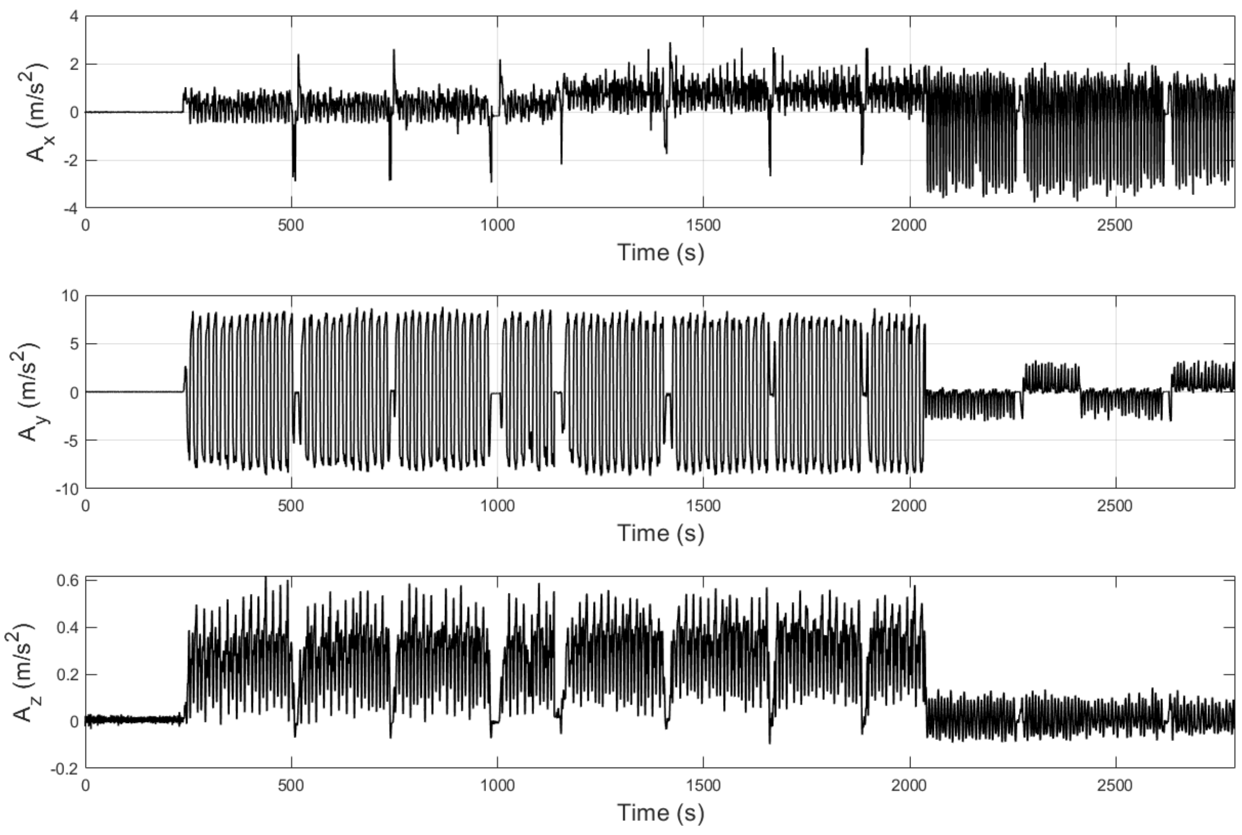


Figure 7.35: Target accelerations for the Calibration trial dataset.

The trajectory of the normalized inputs through the input space, shaded according to the target lateral acceleration, is visualized in Figure 7.36. The lower velocity subset corresponds to the square maneuver portion of the calibration routine, while the higher velocity subset corresponds to the figure-8 maneuvers. The dynamics vary more strongly at higher velocities, as expected. The relationship between the steer angle and the lateral slip remains the same as in the previous three datasets, as expected. The inputs for the calibration routine are similar to what was measured for the figure-8 and square maneuver datasets. This indicates that the GRBFNN should be able to recall the dynamics learned for those maneuvers when tested on the calibration routine.

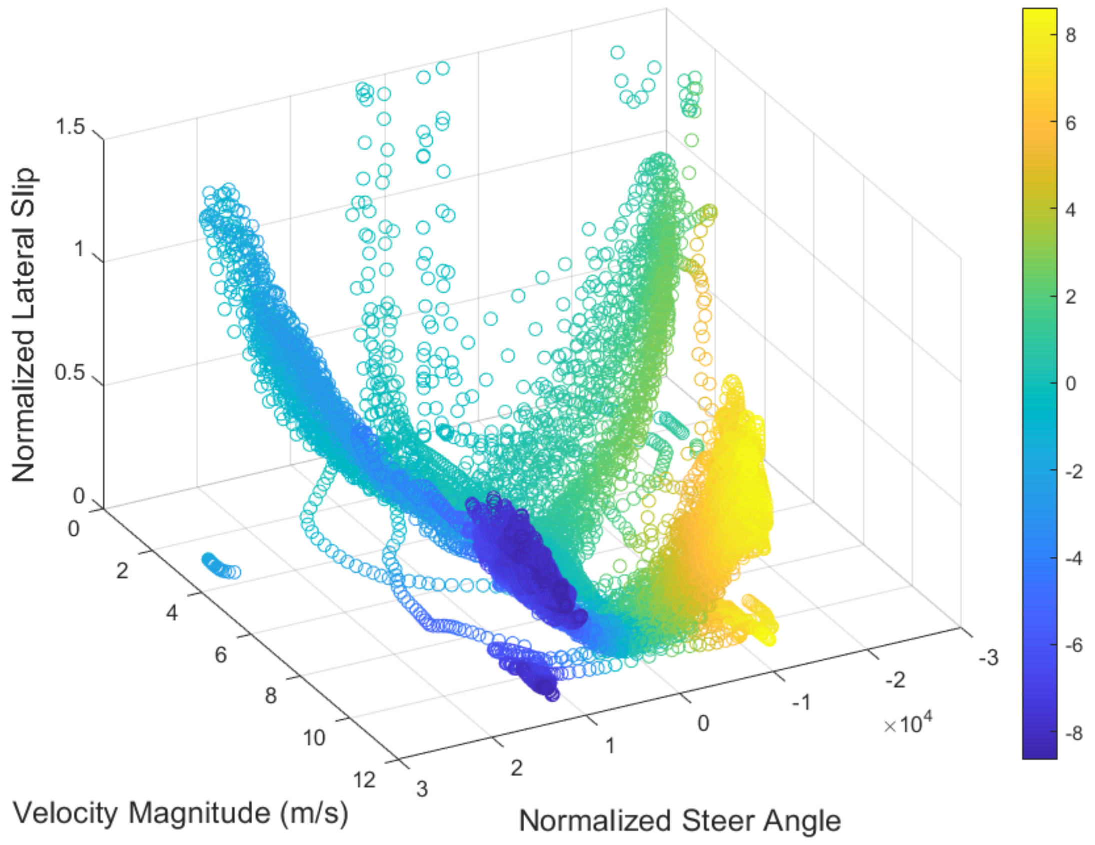


Figure 7.36: The calibration maneuver trajectory in the input feature space shaded according to the target lateral acceleration.

Figure 7.37 confirms that the neural network model was able to estimate the target accelerations to a high degree of accuracy. The magnitude of the estimation error never reaches more than 1.5 m/s^2. The increased error for the second stage of the figure-8 portion of the routine is likely a result of the execution of the maneuver during this stage, which was more aggressive than what was observed during training. This was done to test how the neural network model would respond if the chosen maneuver is performed differently than it as for the training maneuver. The result is a small increase in the maximum error.

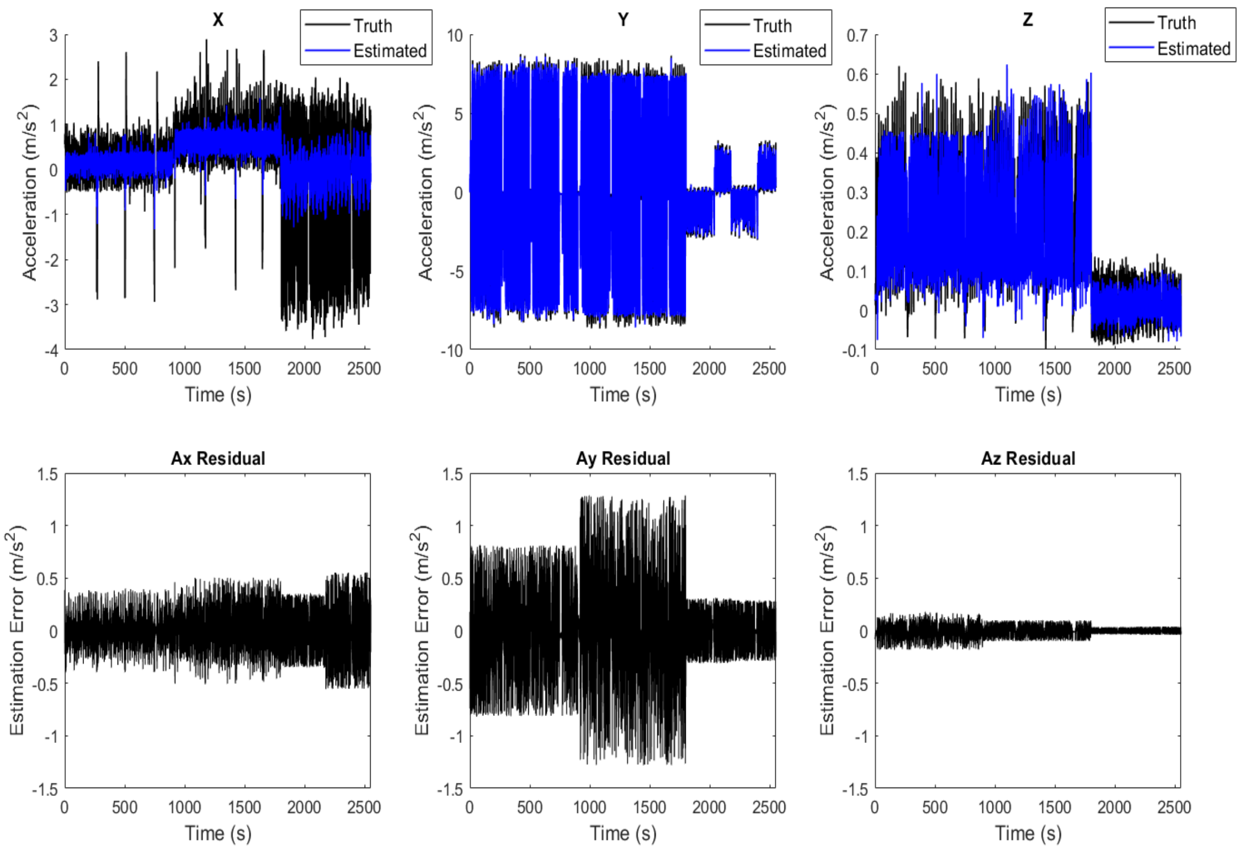


Figure 7.37: Comparison of the true vehicle frame accelerations (Black) to the estimated vehicle frame accelerations (Blue) for the calibration trial, with the residual estimation error.

The results of the IMU emulation for the calibration routine show that the model is able to estimate the vehicle frame accelerations for a combined test routine for which it was previously trained on the independent figure-8 and square maneuver datasets. While the measured errors are on the expected level given what was observed for the figure-8 and square maneuver datasets, the sensitivity test suggests that it is still too large to obtain a pose estimate of the desired accuracy. However, given a greater period of time, the pose error may decrease, as variations due to the residual offset each other in the limit. A characterization of the pose error as a function of the calibration time should resolve the question.

7.5.2 Pose Estimation

The calibration routine in this thesis was designed specifically to maximize the vehicle dynamics in the lateral and longitudinal directions, and hence the observability of the sensor pose. A relatively high accuracy was therefore expected, despite the results of the sensitivity test. To obtain a clearer picture of what the pose estimation process entails, the accelerations induced by the sensor-to-vehicle lever arm were analyzed. It is these induced accelerations that differentiate the vehicle frame from the sensor frame, and therefore need to be accurately captured in order to correctly estimate the sensor lever arm. Figure 7.38 shows the accelerations induced by the lever arm in the vehicle frame. The induced longitudinal accelerations are on the order of 1 m/s^2 at their maximum, while the induced lateral accelerations reach a maximum of 2 m/s^2. The induced vertical accelerations are uniformly smaller than the longitudinal or lateral accelerations, but are much greater for the square maneuver than for the figure-8 maneuver.

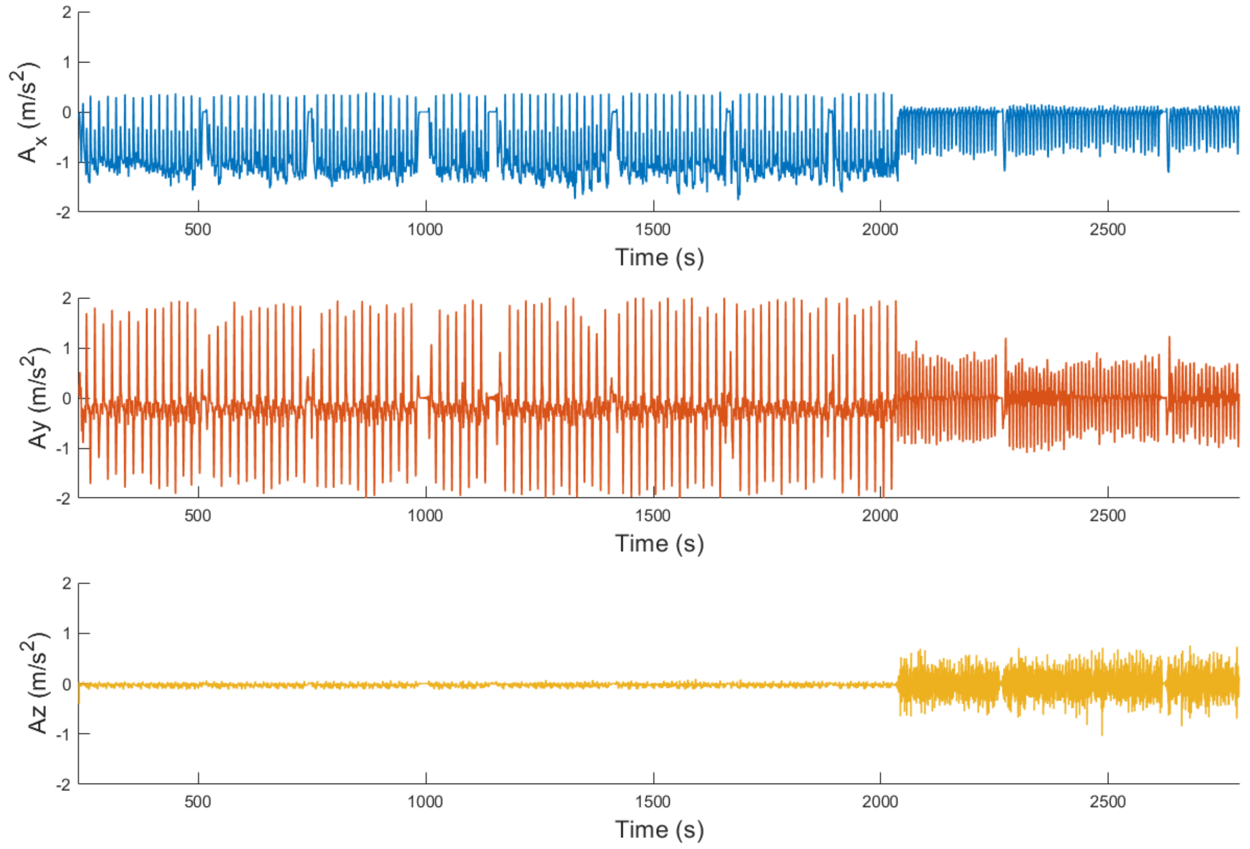


Figure 7.38: The accelerations induced by the IMU-to-Vehicle lever arm for the calibration trial, displayed in the vehicle frame.

The induced accelerations are on a scale that is significant in comparison to the overall vehicle frame accelerations. The 2 m/s^2 induced in the lateral direction are approximately 25% of the total acceleration measured by the ground truth system. The induced longitudinal accelerations are on a similar scale. However, the vertical accelerations are much smaller relative to the total vertical acceleration in the vehicle frame.

These values correspond to the accelerations induced by the full lever arm between the IMU and the vehicle frame centered on the chosen control point. However, the desired estimation accuracy is to within 1 cm of the true sensor position. The differences in the induced accelerations for a relative distance of 1 cm in the longitudinal direction are shown in Figure 7.39. These accelerations are on a much smaller scale than the full induced accelerations, and are dwarfed by the residual estimation error. To estimate the lever arm to this precision would require either a far more accurate model, or some statistical convergence over time. The full transformed and measured sensor frame accelerations are displayed in Figure 7.40. The transformed accelerations match the measured accelerations very well overall, including during the static portions of the dataset.

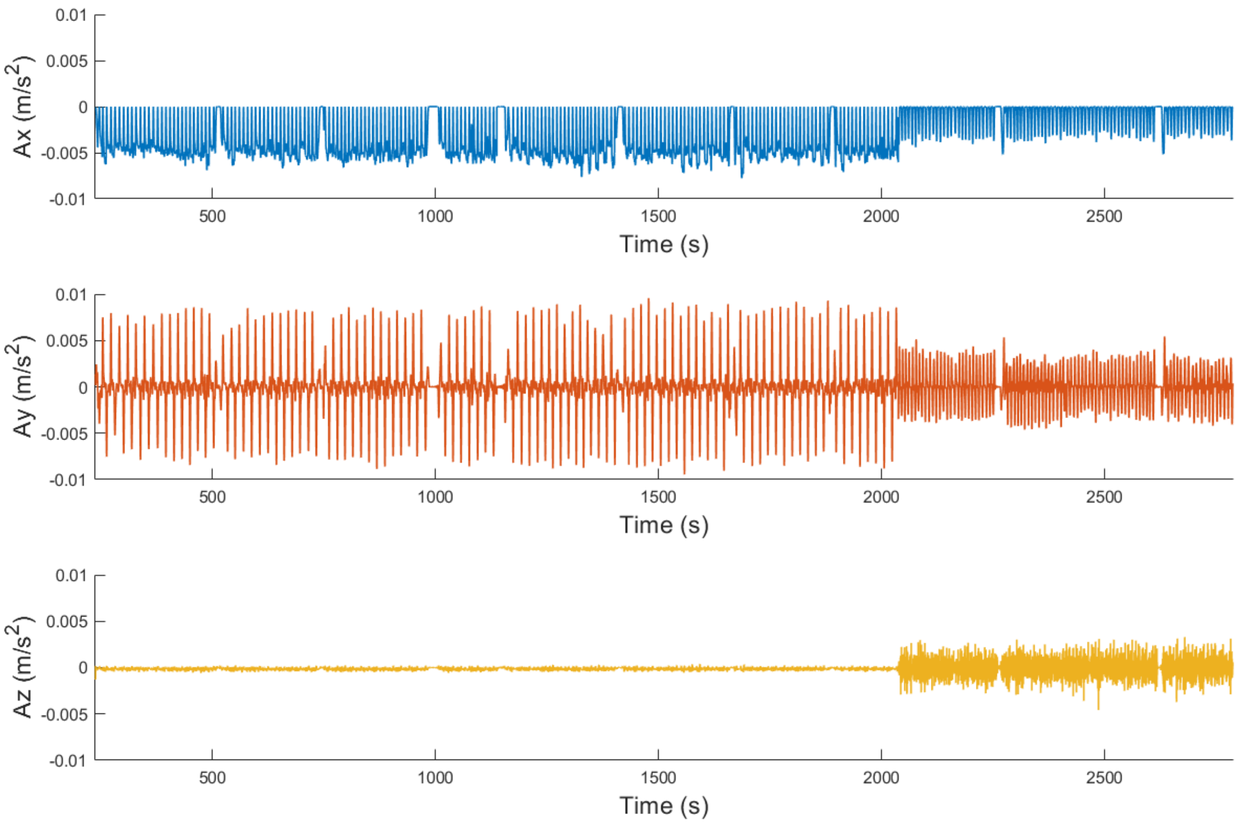


Figure 7.39: The accelerations induced by a 1 cm longitudinal change in the IMU-to-Vehicle lever arm for the calibration trial.

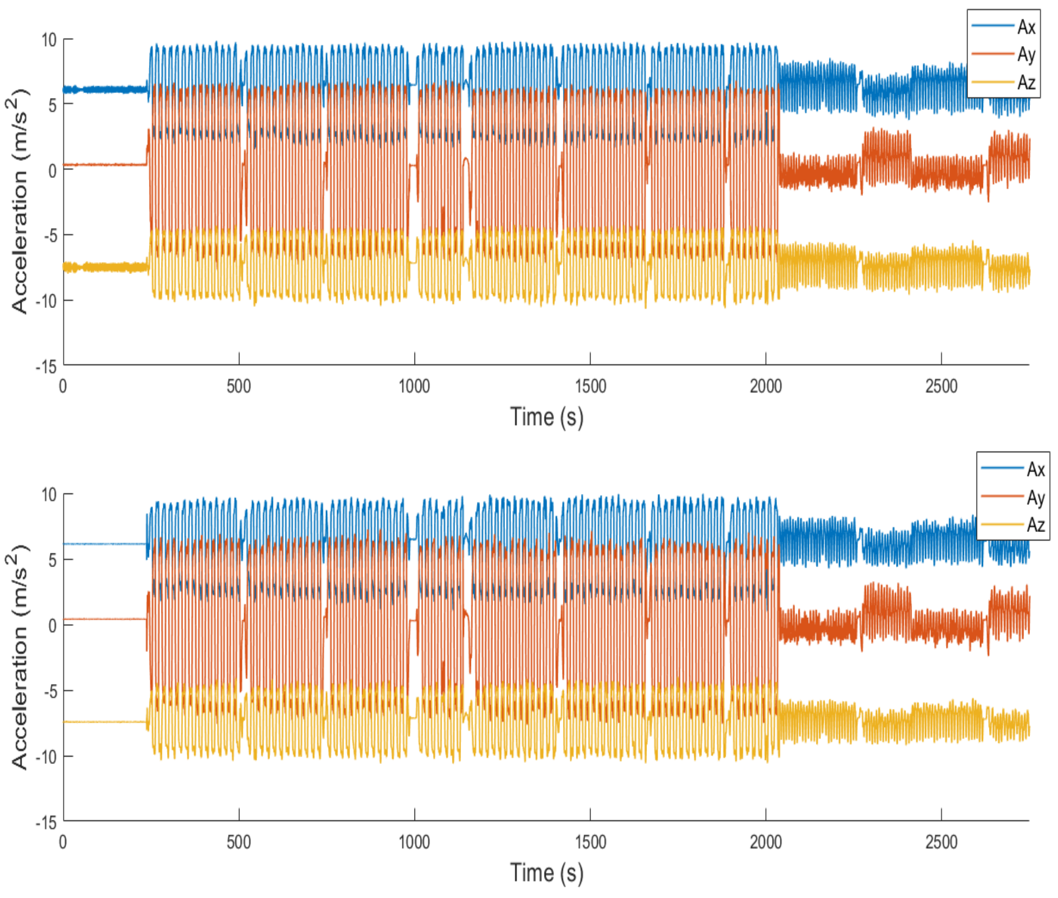


Figure 7.40: All accelerations transformed from the vehicle frame to the sensor frame (above) compared to all accelerations measured in the sensor frame (below) for the calibration trial.

However, the residual error, shown in Figure 7.41, is still on the 1 m/s^2 scale. The signal-to-noise ratio for a 1 cm relative lever arm is well below this. This indicates that it will be very difficult for the optimizer to correctly estimate the sensor position to this scale, unless there is some form of significant statistical convergence.

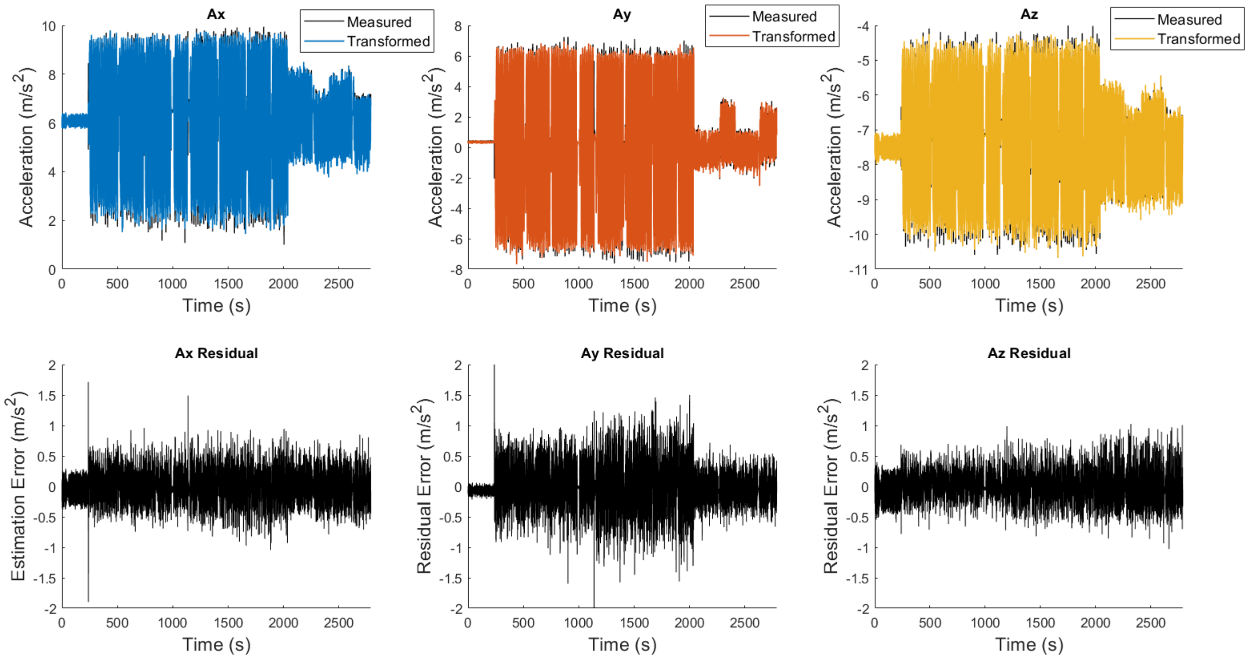


Figure 7.41: Measured sensor frame accelerations compared to estimated accelerations transformed from the vehicle frame for the calibration trial, with residual error.

The pose estimation error was therefore characterized as a function of the calibration time. This analysis serves both to bound the extrinsic calibration error in the limit, as well as to recommend a particular calibration time threshold for a given desired accuracy. The full calibration routine was 45 minutes long. The routine was cut at successive three minute increments to create a set of calibration routines of increasing lengths. These derivative routines are fed through the optimizer to obtain pose estimates for each particular calibration time length. The trajectory of the error naturally depends on which maneuver is conducted first. However, this experiment was enough to roughly characterize each individual component of the pose estimation error as a function of the calibration time.

Figure 7.42 shows how the individual position errors varied with time. The longitudinal and lateral positions converge to within 3 cm of the true position by the end of the calibration routine. After 45 minutes, the error in the longitudinal position is 1.5 cm, while the lateral position error is 2.5 cm. The vertical position estimate, however, never stabilizes. After 45 minutes, the vertical error was 15.5 cm. This suggests that the observability of the vertical lever arm during the calibration routine was too low for it to be accurately estimated.

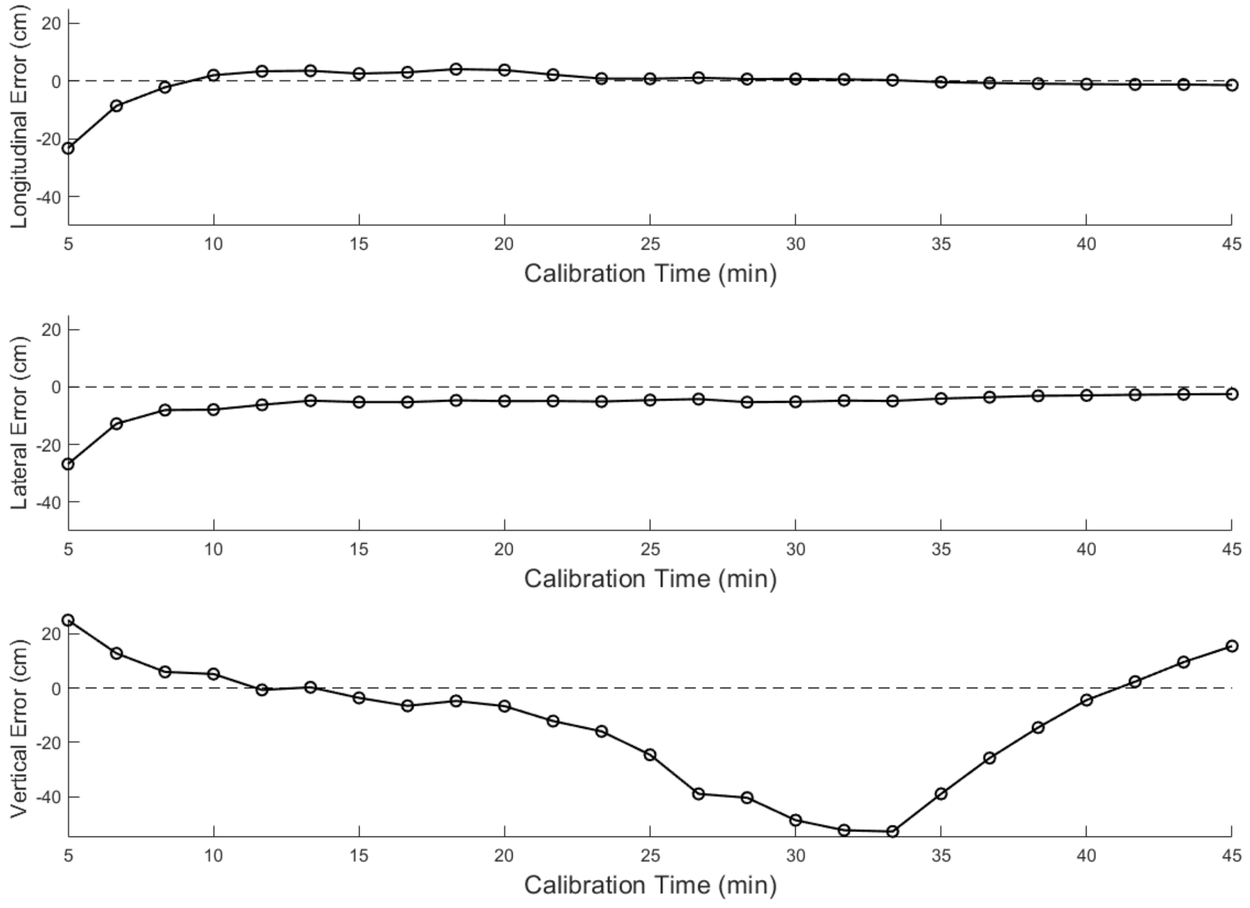


Figure 7.42: Sensor position estimation error as a function of calibration time.

Figure 7.43 shows the variation of the sensor orientation errors with time. All three of the sensor orientation angles converge to the truth by the end of the calibration routine. The roll and pitch angle errors are 0.20 deg and 0.22 deg respectively after 45 minutes. The yaw angle estimate within 0.1 deg of the true yaw angle by the end of the calibration routine. This is likely due to the significant yaw dynamics across the entire calibration routine. The angle estimates appear to approach the truth around the 15 minute mark. However, this is probably due to random variation, rather than statistical convergence, because the estimate diverges afterwards.

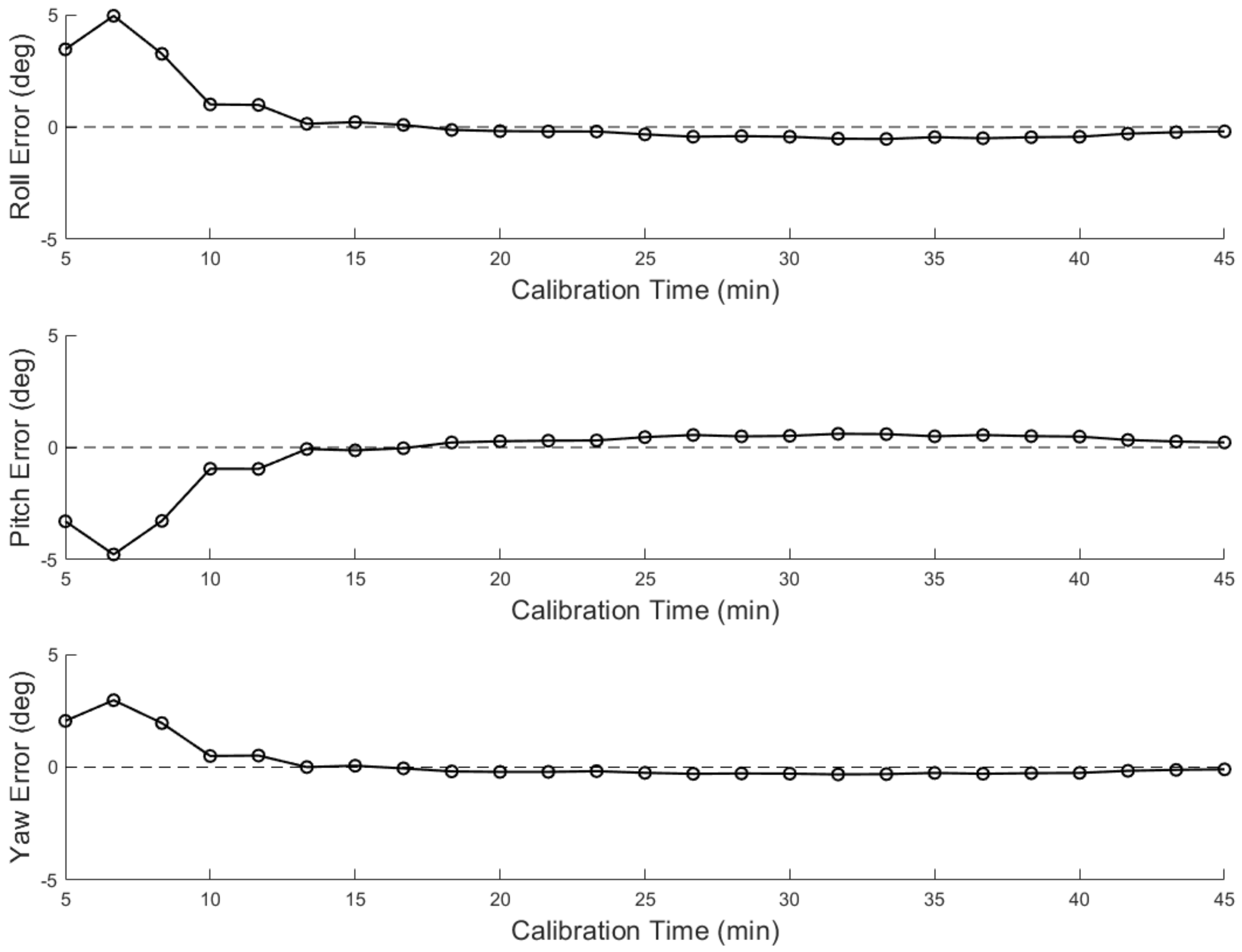


Figure 7.43: Sensor orientation error as a function of calibration time.

The total error in the sensor position is shown in Figure 7.44. The error reaches a minimum around 6 cm at around the 42 minute mark. It is dominated by the error in the vertical lever arm estimate, which failed to converge for the calibration trial, as shown previously. The vertical position, however, is also the least important variable to estimate for modelling or control purposes. If only the lateral and longitudinal position are desired, then the algorithm will perform satisfactorily. Figure 7.45 shows how the total sensor orientation error varies over time. After the full 45 minute calibration routine, the error is 0.3 deg. The total orientation estimation error remains bounded below 2.0 deg after the 10 minute mark.

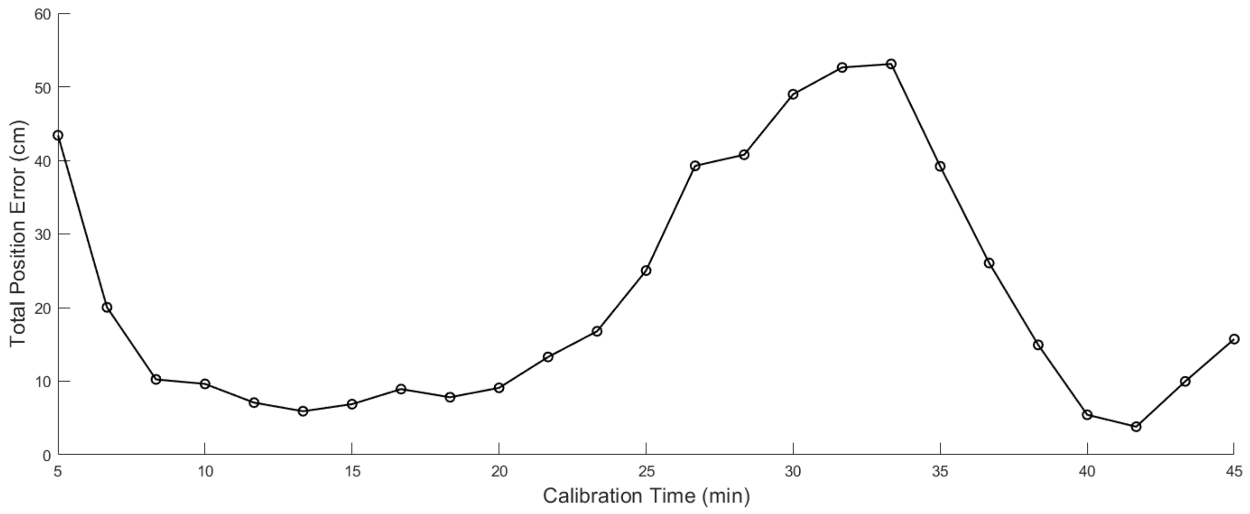


Figure 7.44: Total position error as a function of calibration time

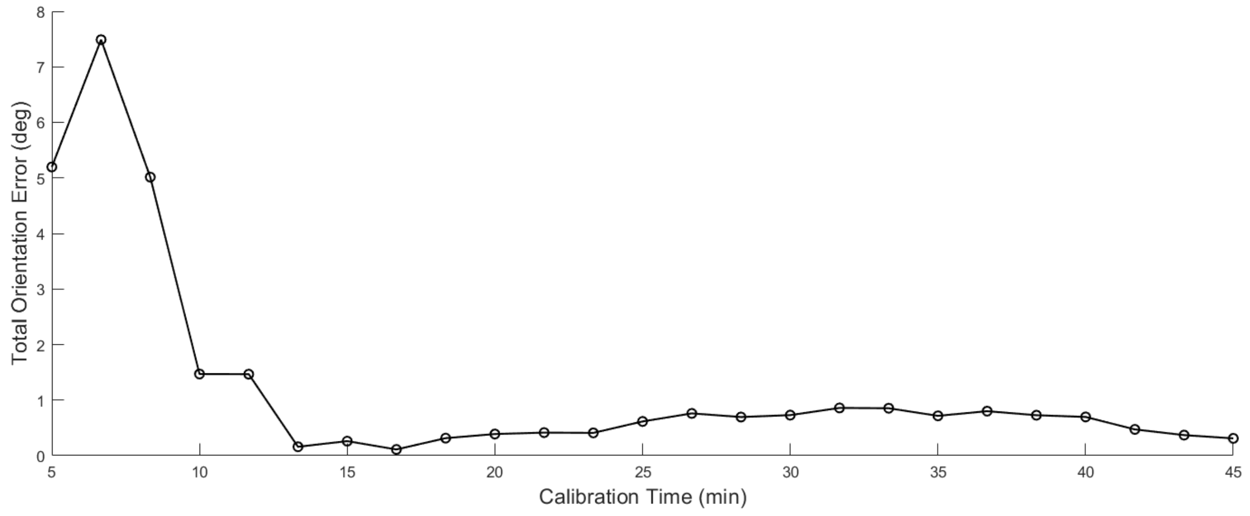


Figure 7.45: Total orientation error as a function of calibration time.

The results from the calibration trial show that the algorithm can be successfully applied on a calibration routine once the model has been trained on maneuvers similar to, but different from, those executed during that routine. The IMU position estimation error was 1.46 cm in the longitudinal direction, 2.49 cm in the lateral direction, and 15.46 cm in the vertical direction at the end of the calibration routine. The IMU orientation estimation error was 0.20 degrees in pitch, 0.22 degrees in roll, and 0.10 degrees in yaw. Despite the low signal-to-noise ratio for the induced accelerations when the lever arm is on the centimeter scale, the algorithm was able to estimate the lateral and longitudinal lever arms to within 3 cm by the end of the calibration routine, and to within 0.25 deg in each of the three orientation angles. The only component of the sensor pose for which the estimate did not converge after 45 minutes was the vertical position.

Chapter 8

Conclusions and Future Work

This thesis has developed a direct IMU-to-Vehicle extrinsic calibration routine that can be performed both autonomously and online. The calibration required only a minimal array of sensors and a test maneuver. The choice of the maneuver was also relatively free; the only requirements were that it contain a standstill phase and a dynamic phase. The calibration was accomplished using a two-stage approach: first a gaussian radial basis function neural network was used to emulate the output of an inertial measurement unit aligned with and centered on the origin of the vehicle frame. Then a maximum likelihood based search method estimated the pose of the IMU relative to this emulated IMU.

The Gaussian Radial Basis Function Neural Network model was able to estimate the linear accelerations to a high degree of accuracy both in simulation and in experimental validation, while only training on a limited dataset. The neural network showed strong generalization properties to datasets of varying forward velocity and steer angle profile, and was capable of training and estimating in noisy measurement environments while rejecting the noise to a near-perfect degree. A cross-validation algorithm successfully prevented the neural network from over-fitting the training data. The input feature normalization process enables it to further generalize from a limited amount of training data. With only a few training maneuvers, the model was able to learn the vehicle dynamics, and therefore emulate an IMU to a high degree of accuracy.

Given estimates of the linear accelerations in the vehicle frame from the IMU emulation algorithm, the maximum likelihood search method was used to precisely estimate the pose of the inertial measurement unit. The maximum likelihood acted as a statistical estimator of the sensor pose, enabling a covariance-matrix adaptation based optimizer to find the true sensor position to within 2 mm and the true sensor orientation to within 0.02 degrees for all simulated datasets. The calibration algorithm also performed well on experimental data. For a calibration trial combining figure-8 and square maneuvers, the true sensor position was estimated to within 3 cm in the lateral and longitudinal directions. The true sensor orientation, meanwhile, was estimated to within 0.25 deg in each direction. The only element of the sensor pose that failed to converge during the calibration trial was the vertical sensor position.

In the future, some sources of error could be addressed, allowing for a more accurate estimate of the sensor's extrinsic calibration parameters. A major potential source of error is the bias instability of the calibrated sensor. A constant bias estimate was used in this thesis, but a more sophisticated bias estimation process could eliminate some of the resulting model error. Another source of error encountered during the experimental trials, and the road drive in particular, was the gravity vector. On roads with non-zero grade, the gravity vector shifted away from true vertical relative to the vehicle. This altered the accelerations measured by the ground truth system away from the true vehicle frame accelerations. On flat ground, this is not an issue. However, the calibration algorithm will inevitably be used on sloped or banked ground. It will therefore become a priority in the future to ensure that it performs as well in these environments as it does on flat ground. This issue could potentially be resolved with the addition of a pitch and roll model that corrects for the shift in the gravity vector.

Another potential avenue to expand the operational viability of the calibration algorithm is to train the neural network model on a variety of road surface conditions. The relationship between the lateral slip and the steering angle changes with the frictional properties of the road surface. By training on wet roads, dirt roads, or degraded surfaces, the neural network will learn to model the dynamics in any of these environments. The algorithm could then be applied, no matter the external conditions, without loss of performance.

A wide array of different techniques exist to extrinsically calibrate an inertial measurement unit to other sensors that are commonly used on autonomous vehicles. Once the IMU has been calibrated relative to the vehicle frame, all of the other sensors may be calibrated relative to the IMU. This opens up the possibility for a routine that is capable of extrinsically calibrating the entire sensor suite of an autonomous vehicle online, with only minimal human intervention. This would enable the operator to rearrange sensors at will. If a sensor is knocked out of alignment, the autonomy of the vehicle would no longer be threatened. Time and labor intensive manual calibration routines would also no longer be necessary each time a vehicle is launched from the factory floor, or whenever maintenance is required.

The development of a full sensor suite calibration routine is a real possibility in the immediate future. This would entail the integration of the IMU-to-Vehicle calibration routine that has been developed in this work with a number of other IMU-to-Sensor calibration algorithms, including IMU-to-LiDAR and IMU-to-Camera calibrations that involve visual odometry. These could potentially be performed after the IMU-to-Vehicle calibration algorithm has finished running. This would give the perception system calibrations access to the accelerations and angular rates in the vehicle frame in addition to those measured by the physical IMU on board the vehicle. The full sensor suite calibration routine could be implemented and integrated into the software of a commercial self-driving car.

However, the calibration routine developed in this thesis is not inherently limited to automobiles. There is a latent potential for this method to be applied to unmanned aerial vehicles as well. GPS-derived velocities and angular rates can be obtained just as easily for an aerial vehicle as they can for a ground vehicle. The greatest hurdle to overcome in this endeavor would be to perform the calibration without a measurement of the vehicle's forward velocity from a wheel odometry system.

The stand-alone IMU extrinsic calibration could also be fused with stand-alone LiDAR extrinsic calibration to enhance the overall performance of the pose estimation. The IMU-to-LiDAR pose estimate would act as an additional constraint on the poses of the IMU and the LiDAR relative to the vehicle frame, enabling a higher degree of precision in the final combined pose estimate. In a similar vein, the addition of a second IMU could allow both IMUs to be calibrated simultaneously. The pose estimates of each IMU relative to the vehicle frame, in addition to the IMU-to-IMU relative pose, could be fused using a Kalman filter to produce an even more refined combined estimate. It would be useful to analyze the marginal benefit of incorporating each additional IMU to the calibration.

Improving the accuracy and robustness of the described calibration method will remain a priority in the future. It will be important to ensure that the neural network model can emulate the vehicle frame accelerations correctly for any physically conceivable maneuver. This will necessarily entail a rigorous analysis of the quantity of training data required to fully train the model. This might be accomplished by assessing the density of training data points within regions of the input feature space. It will be useful to compile a set of training maneuvers that is sufficient to build a complete model of the vehicle accelerations.

Another related issue is that the neural network currently needs to be trained for the specific model of vehicle that it will be deployed on. This issue could be addressed by making the IMU emulation model 'vehicle agnostic'. Ground vehicles possess similar dynamic models. A base model equipped with a set of vehicle parameters could be retrained to model the dynamics of another vehicle. This would eliminate the issue of limited training data, and would further reduce the manual effort required to employ the calibration routine. Finally, the dynamic vehicle model that is a byproduct of the calibration method developed in this thesis could be used for any number of sensor fusion, estimation, or localization tasks.

Bibliography

[1] Defense Advanced Research Projects Agency (DARPA). “Grand Challenge 2004 Final Report”. In: (2004).

[2] H. Akeike. “Information Theory and an Extension of the Maximum Likelihood Principle”. In: *Proceedings of the 2nd International Symposium on Information Theory* (1971).

[3] U. Akram. “An Improved Pi-Sigma Neural Network with Error Feedback for Physical Time Series Prediction”. In: *International Journal of Advanced Trends in Computer Science and Engineering* 8 (2019), pp. 276–284.

[4] U. Akram, R. Ghazali, and M. Mushtaq. “A Comprehensive Survey on Pi-Sigma Neural Network for Time Series Prediction”. In: *Journal of Telecommunication, Electronic, and Computer Engineering* 9 (2017), pp. 57–62.

[5] J. Almazán et al. “Full Auto-Calibration of a Smartphone on Board a Vehicle using IMU and GPS Embedded Sensors”. In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. 2013, pp. 1374–1380.

[6] S. Amari. “Statistical Theory of Overtraining - Is Cross-Validation Asymptotically Effective”. In: *Advances in Neural Information Processing Systems* 8 (1996), pp. 176–182.

[7] K.J. Åström and P. Eykhoff. “System Identification — A Survey”. In: *Automatica* 7.2 (1971), pp. 123–162. ISSN: 0005-1098.

[8] Guilherme Barreto and Luis Mota Souza. "Adaptive filtering with the self-organizing map: A performance comparison". In: *Neural networks : the official journal of the International Neural Network Society* 19 (July 2006), pp. 785–98. DOI: `10.1016/j.neunet.2006.05.005`.

[9] A.R. Barron. "Universal Approximation Bounds for Superpositions of a Sigmoidal Function". In: *IEEE Transactions on Information Theory* 39 (1993), pp. 930–945.

[10] E. Bas, E. Egrioglu, and O. Karahasan. "A Pi-Sigma Artificial Neural Network based on Sine-Cosine Optimization Algorithm". In: *Granular Computing* (2021), pp. 1–8.

[11] N. Benoudjit et al. "Width optimization of the Gaussian kernels in Radial Basis Function Networks." In: *Proceedings of the 10th European Symposium on Artificial Neural Networks.* Jan. 2002, pp. 425–432.

[12] G.E.P. Box and D. Pierce. "Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models". In: *Journal of the American Statistical Association* 65 (1970), pp. 1509–1526.

[13] J.S. Bridle. "Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters". In: *Advances in Neural Information Processing Systems* 2 (1990), pp. 211–217.

[14] D. Broomhead and D. Lowe. "Multivariable Functional Interpolation and Adaptive Networks". In: *Complex Systems* 2 (1988), pp. 321–355.

[15] A. Brown and R.L. Smith. *Automotive Sensor Alignment with External IMU Tool.* US Patent No. 9568592B1. 2017.

[16] H. Bruce. "Modeling and Simulation of GPS Multipath Propagation". PhD thesis. Queensland University of Technology, 2001.

[17] J. Campos, D. Hendry, and H.M. Krolzig. "Consistent Model Selection by an Automatic Gets Approach". In: *Oxford Bulletin of Economics and Statistics* 65 (2003), pp. 803–819.

[18] M. Casdagli. “Nonlinear Prediction of Chaotic Time Series”. In: *Physica* 35 (1989), pp. 335–356.

[19] David Cassel. *Remembering Shakey, the First Intelligent Robot.* Accessed July 2023. URL: `https://thenewstack.io/remembering-shakey-first-intelligent-robot/`.

[20] J. Castle, J.A. Doornik, and D. Hendry. “Evaluating Automatic Model Selection”. In: *Journal of Time Series Econometrics* 3 (2011).

[21] J. Castle and D. Hendry. “A Low-Dimension Portmanteau Test for Nonlinearity”. In: *Journal of Econometrics* (2010).

[22] Q. Chen, Q. Zhang, and X. Niu. “Estimate the Pitch and Heading Mounting Angles of the IMU for Land Vehicular GNSS/INS Integrated System”. In: *IEEE Transactions on Intelligent Transportation Systems* (2020).

[23] S. Chen, S. Billings, and P. Grant. “Nonlinear System Identification using Neural Networks”. In: *International Journal of Control* 51 (1990), pp. 1191–1214.

[24] *Concept of directional optimization in CMA-ES algorithm.* Accessed July 2023. URL: `https://upload.wikimedia.org/wikipedia/commons/d/d8/Concept_of_directional_optimization_in_CMA-ES_algorithm.png`.

[25] SenNav Corporation. *The Difference between a Ring-Laser Gyro, and Fiber-Optic Gyro.* Accessed March 2022. URL: `https://sennavs.com/the-difference-between-ring-laser-gyro-and-fiber-optic-gyro/`.

[26] VectorNav Corporation. *High Performance Gyroscopes.* Accessed March 2022. URL: `https://www.vectornav.com/resources/inertial-navigation-primer/theory-of-operation/theory-gyros`.

[27] T. Cover. “Geometrical and Statistical Properties of Systems with Linear Inequalities, with Applications in Pattern Recognition”. In: *IEEE Transactions on Electronic Computers* 3 (1965), pp. 326–334.

[28] F. Cuesta and A. Ollero. *Intelligent Mobile Robot Navigation.* Berlin, Heidelberg: Springer-Verlag, 2005.

[29] Georgi Dalakov. *History of Computers and Computing Automata, Jacques Vaucanson @ONLINE.* 2022. URL: `http://history-computer.com/Dreamers/Vaucanson.html`.

[30] Kaushik Das. *HOW RECURRENT NEURAL NETWORK (RNN) WORKS.* Accessed July 2023. URL: `https://dataaspirant.com/how-recurrent-neural-network-rnn-works/`.

[31] J.A. Doornik. "Encompassing and Automatic Model Selection". In: *Oxford Bulletin of Economics and Statistics* 70 (2008), pp. 915–925.

[32] S. El Hihi and Y. Bengio. "Hierarchical Recurrent Neural Networks for Long Term Dependencies". In: *Advances in Neural Information Processing Systems* 8 (1996), pp. 493–499.

[33] P. Frasconi and M. Gori. "Computational Capabilities of Local-feedback Recurrent Networks acting as Finite-state Machines". In: *IEEE Transactions on Neural Networks* 7 (1996), pp. 1521–1524.

[34] B. Friedlander. "The Overdeterimed Recursive Instrumental Variable Method". In: *IEEE Transactions on Automation and Control* 29 (1984), pp. 353–356.

[35] Q. Fu et al. "Autonomous In-motion Alignment for Land Vehicle Strapdown Inertial Navigation System without the Aid of External Sensors". In: *The Journal of Navigation* 71.6 (June 2018), pp. 1312–1328.

[36] J. Furgale P. Rehder and R. Siegwart. "Unified Temporal and Spatial Calibration for Multi-Sensor Systems". In: *IEEE International Workshop on Intelligent Robots and Systems (IROS)* (2013).

[37] Erran Gat. "Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots". In: *Proceedings of the Tenth National Conference on Artificial Intelligence*. AAAI Press, 1992, pp. 809–815.

[38] M. Gerdts. *The Single Track Model*. University of Bayreuth, 2003.

[39] Sean Gerrish. *How Smart Machines Think*. Cambridge, MA: MIT Press, 2018.

[40] C.L. Giles and B.G. Horne. "Representation of Learning in Recurrent Neural Network Architectures". In: *Proceedings of the 8th Yale Workshop on Adaptive and Learning Systems* (1994), pp. 128–134.

[41] M. Gori and A. Tesi. "On the Problem of Local Minima in Backpropagation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (1992), pp. 76–86.

[42] Azidine Guezzaz et al. "A Multilayer Perceptron Classifier for Monitoring Network Traffic". In: *Big Data and Networks Technologies*. Ed. by Yousef Farhaoui. 2020, pp. 262–270.

[43] G. Hager. "Task-Directed Sensor Fusion and Planning: A Computational Approach". In: *The Kluwer International Series in Engineering and Computer Science*. Vol. 99. Berlin, Heidelberg: Springer-Verlag, 1990.

[44] Youssef Harkouss, Mcheik Souhad, and Roger Achkar. "Accurate Wavelet Neural Network for Efficient Controlling of an Active Magnetic Bearing System". In: *Journal of Computer Science* (Jan. 2010). DOI: 10.3844/jcssp.2010.1452.1459.

[45] E.J. Hartman, J.D. Keeler, and J.M. Kowalski. "Layered Neural Networks with Gaussian Hidden Units as Universal Approximators". In: *Neural Computation* 2 (1990), pp. 210–215.

[46] K. Hausman et al. "Self-Calibrating Multi-Sensor Fusion with Probabilistic Measurement Validation for Seamless Sensor Switching on a UAV". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 4289–4296.

[47] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, 1999. ISBN: 0-13-273350-1.

[48] D. Hendry and H.M. Krolzig. "The Properties of Automatic Gets Modeling". In: *The Economic Journal* 115 (2005), pp. 32–61.

[49] J. Hesterman. "Maximum Likelihood Estimation with a Contracting Grid Search Algorithm". In: *IEEE Transactions on Nuclear Science* 57.3 (2010), pp. 1077–1084.

[50] S. Hochschreiter and J. Schmidhuber. "LSTM can solve Hard Long Time-lag Problems". In: *Advances in Neural Information Processing Systems* 9 (1997), pp. 473–479.

[51] B. Hollerbach. *Robot Motion: Planning and Control*. Cambridge, MA: MIT Press, 1982.

[52] Joseph Hooper. *From DARPA Grand Challenge 2004: DARPA's Debacle in the Desert @ONLINE*. 2004. URL: `https://www.popsci.com/scitech/article/2004-06/darpa-grand-challenge-2004darpas-debacle-desert/`.

[53] H. Horton. *Seven Reasons Your Life Depends on an Accurate IMU in a Self-Driving Car*. Accessed March 2022. URL: `https://medium.com/@mikehorton/7-reasons-your-life-depends-on-an-accurate-imu-inertial-measurement-unit-in-a-self-driving-car-75298d5cff9e`.

[54] W. Ilewicz and A. Nawrat. "Advanced Technologies for Intelligent Systems of National Border Security". In: Springer, 2013. Chap. 13: Direct Method of IMU Calibration.

[55] H. Inoue. "Force feedback in precise assembly tasks". In: (1974).

[56] A. Jain, L. Zhang, and L. Jiang. *High-Fidelity Sensor Calibration for Autonomous Vehicles*. Accessed March 2022. URL: `https://medium.com/wovenplanetlevel5/high-fidelity-sensor-calibration-for-autonomous-vehicles-6af06eba4c26`.

[57] A.K. Jain. “Real-Time Object Measurement and Classification”. In: *NATO ASI Series*. Vol. 42. Berlin, Heidelberg: Springer-Verlag, 1988.

[58] S. Kern, S. Muller, and N. Hansen. “Learning Probability Distributions in Continuous Evolutionary Algorithms - A Comparative Review”. In: *Natural Computing* 2004 (3 2004), pp. 77–112.

[59] D. Kim, S. Shin, and I. Kweon. “On-Line Initialization and Extrinsic Calibration of an Inertial Navigation System With a Relative Preintegration Method on Manifold”. In: *IEEE Transactions on Automation Science and Engineering* 15.3 (2018), pp. 1272–1285.

[60] S.F. King, R.C. Gonzalez, and C.S.G. Lee. *Robotics, Control, Sensing, Vision, and Intelligence*. USA: McGraw-Hill, Inc., 1987.

[61] S. Kullback and R.A. Leibler. “On Information and Sufficiency”. In: *Annals of the Institute of Statistics and Mathematics* 22 (1951), pp. 79–86.

[62] J. Lee, D. Hanley, and T. Bretl. “Extrinsic Calibration of Multiple Inertial Sensors from Arbitrary Trajectories”. In: *IEEE Robotics and Automation Letters* 7.2 (2022).

[63] W.A. Light. “Some Aspects of Radial Basis Function Approximation”. In: *Approximation Theory, Spline Functions, and Applications* 256 (1992), pp. 163–190.

[64] Z. Lin and M.B. Beck. “On the Identification of Model Structure in Hydrological and Environmental Systems”. In: *Water Resources Research* 43 (2007).

[65] D. Lowe. “Adaptive Radial Basis Function Nonlinearities, and the Problem of Generalization”. In: *First IEEE International Conference on Artificial Neural Networks* (1989), pp. 171–175.

[66] Kaushik Mani. *GRU’s and LSTM’s*. Accessed July 2023. URL: `https://towardsdatascience.com/grus-and-lstm-s-741709a9b9b1`.

[67] R. Marco et al. “A Novel IMU Extrinsic Calibration Method for Mass Production Land Vehicles”. In: *IEEE Sensors Journal* (2021).

[68] F.L. Markley. “Attitude Determination using Vector Observations and the Singular Value Decomposition”. In: *Journal of the Astronautical Sciences* 36.3 (1988), pp. 245–258.

[69] D.Q. Mayne. “A Method for Estimating Discrete Time Transfer Functions”. In: *Advances in Computer Control* (1967).

[70] James D. Mccaffrey. *Graphing a Gaussian Kernel in 3-D*. Accessed July 2023. URL: https://jamesmccaffrey.wordpress.com/2014/01/25/graphing-the-gaussian-kernel-function-in-3-d/.

[71] J. McCarthy and P. Hayes. “Some Philosophical Problems from the Standpoint of Artificial Intelligence”. In: *Machine Intelligence 4*. Ed. by B. Meltzer and D. Michie. Edinburgh University Press, 1969, pp. 463–502.

[72] Douglas McGray. *The Great Robot Race*. Accessed March 20th, 2022. URL: https://www.wired.com/2004/03/robot-3/.

[73] Puneeth Meruva. *Sensor Calibration is Critical to the Future of Automated Vehicles*. Accessed July 2023. URL: https://www.trucks.vc/blog/sensor-calibration-is-critical-to-the-future-of-automated-vehicles.

[74] N. Montalbano and T.E. Humphreys. “A Comparison of Methods for Online Lever Arm Estimation”. In: *Proceedings of the ION PLANS Meeting* (2018).

[75] Michael Montemerlo et al. “Winning the DARPA Grand Challenge with an AI Robot.” In: Jan. 2006.

[76] J. Moore and H. Weiss. “Recursive Prediction Error Methods for Adaptive Estimation”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9 (1979), pp. 197–205.

[77] H.P. Moravec. *Robot Rover Visual Navigation*. Ann Arbor, MI: UMI Research Press, 1981.

[78] A.J. Nevins. “A human oriented logic for automatic theorem-proving”. In: *Journal of the ACM (JACM)* 21.4 (1974), pp. 606–621.

[79] Mark J.L. Orr. “Introduction to Radial Basis Function Networks”. In: (1996).

[80] J. Park and I. Sandberg. “Approximation and Radial Basis Function Networks”. In: *Neural Computation* 5 (2 1993), pp. 305–316.

[81] J. Park and I.W. Sandberg. “Universal Approximation using Radial Basis Function Networks”. In: *Neural Computation* 3 (1991), pp. 246–257.

[82] T. Poggio and F. Girosi. “Networks for Approximation and Learning”. In: *Proceedings of the IEEE* 78 (1990), pp. 1481–1497.

[83] P. Polack. “The Kinematic Bicycle Model: A Consistent Model for Planning Trajectories for Autonomous Vehicles?” In: *IEEE Intelligent Vehicles Symposium* (2017).

[84] M.J.D. Powell. “Radial Basis Function Approximation to Polynomials”. In: *Numerical Analysis* (1988), pp. 223–241.

[85] M.J.D. Powell. “Radial Basis Functions for Multivariable Interpolation”. In: *proceedings of the IMA Conference on Algorithms for the Approximation of Functions and Data* (1985), pp. 143–167.

[86] A. Qormemeti. “Wind Speed Forecasting for Power Generation using a Self-Assembling Closed-loop Recurrent Neural Network”. MA thesis. Houston, TX: Rice University, 2018.

[87] Archit Rastogi. *Two to Four: Bicycle model for Car*. Accessed July 2023. URL: `https://archit-rstg.medium.com/two-to-four-bicycle-model-for-car-898063e87074`.

[88] J. Rehder et al. “Extending kalibr: Calibrating the Extrinsics of Multiple IMUs and of Individual Axes”. In: *2016 IEEE International Conference on Robotics and Automation*. 2016, pp. 4304–4311.

[89] O. Reiersol. “Confluence Analysis by Means of Lag Moments and Other Methods of Confluence Analysis”. In: *Econometrica* 9 (1941), pp. 1–24.

[90] D. Schramm. *Vehicle Dynamics*. Berlin, Heidelberg: Springer-Verlag, 2014. DOI: 10.1007/978-3-540-36045-2_10.

[91] N. Seegmiller, F. Rogers-Marcovitz, and A. Kelly. “Online Calibration of Vehicle Powertrain and Pose Estimation using Integrated Dynamics”. In: *IEEE International Conference on Robotics and Automation* (2012), pp. 3969–3974.

[92] A. Sherstinsky and R.W. Picard. “On the Efficiency of the Orthogonal Least Squares Training Method for Radial Basis Function Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 7.1 (Jan. 1996), pp. 195–200.

[93] Y. Shin and J. Ghosh. “The Pi-Sigma Network: an Efficient Higher Order Neural Network for Pattern Classification and Function Approximation”. In: *IJCNN Seattle International Joint Conference on Neural Networks* 91 (1991).

[94] B. Singh, K. Verma, and A. Thoke. “Approximation and Radial Basis Function Networks”. In: *International Journal of Computer Applications* 116 (19 2015).

[95] J. Sjoberg. “Nonlinear Black-box Modeling in System Identification: A Unified Overview”. In: *Automatica* 31 (1995), pp. 1691–1724.

[96] T. Soderstrom, P. Stoica, and Trulsson E. “Instrumental Variable Methods for Closed-Loop Systems”. In: *10th Annual IFAC World Congress* (1987).

[97] N.A. Spielberg et al. “Neural Network Vehicle Models for High-Performance Automated Driving”. In: *Science Robotics* 4.28 (2019).

[98] M.H. Stone. “The generalized Weierstrass approximation theorem”. In: *Mathematics Magazine* 21.5 (1948), pp. 237–254.

[99] R.S. Sutton. “Two Problems with Back-propagation and Other Steepest Descent Learning Procedures for Networks”. In: *Proceedings of the 8th Annual Conference of the Cognitive Science Society* (1986), pp. 823–831.

[100] C. Urmson. “Autonomous Driving in Urban Environments: Boss and the Urban Challenge”. In: *The DARPA Urban Challenge, Springer Tracts in Advanced Robotics* 56 (2009).

[101] S. Van Huffel and J. Vandewalle. “Comparison of Total Least Squares and Instrumental Variable Methods for Parameter Estimation of Transfer Function Models”. In: *International Journal of Control* 50 (1989), pp. 1039–1056.

[102] G. Wabha. “A Least Squares Estimate of Satellite Attitude”. In: *SIAM Review* 7.3 (1965), p. 409.

[103] B. Walczak and D. L. Massart. “Local Modelling with Radial Basis Function Networks”. In: *Chemometrics and Intelligent Laboratory Systems* 50.2 (2000), pp. 179–198. ISSN: 0169-7439.

[104] L. Wang and H. Garnier. *System Identification, Environmental Modelling, and Control System Design*. Springer, 2012. ISBN: 978-1-4471-5845-5.

[105] K.Y. Wong and E. Polak. “Identification of Linear Discrete Time Systems using the Instrumental Variable Approach”. In: *IEEE Transactions on Automation and Control* 12 (1967), pp. 707–718.

[106] W. Wrigley. “History of Inertial Navigation”. In: *Navigation* 24.1 (1977), pp. 1–6.

[107] Y. Wu. “Versatile Land Navigation using Inertial Sensors and Odometry, In-Motion Alignment and Positioning”. In: *Proceedings of the 2014 DGON Inertial Sensors and Systems* 16.17 (2014), pp. 1–19.

[108] Y. Wu et al. "Using Radial Basis Function Networks for Function Approximation and Classification". In: *ISRN Applied Mathematics* 2012 (Mar. 2012).

[109] Y. Xia and J. Wang. "A Recurrent Neural Network for Nonlinear Convex Optimization Subject to Nonlinear Inequality Constraints". In: *IEEE Transactions on Circuits and Systems* 51 (2004), pp. 1385–1394.

[110] P.C. Young. "A General Theory of Modeling for Badly Defined Dynamic Systems". In: *Modeling, Identification, and Control in Environmental Systems* (1978).

[111] P.C. Young. "An Instrumental Variable Method for Real-Time Identification of a Noisy Process". In: *Automatica* 6 (1970), pp. 271–287.

[112] P.C. Young. "Data Based Mechanistic Modeling of Environmental, Ecological, Economic, and Engineering Systems". In: *Environmental Modeling and Software* 12 (1998), pp. 105–122.

[113] P.C. Young. "Identification of Nonlinear Stochastic Systems by State-Dependent Parameter Estimation". In: *International Journal of Control* 74 (2001), pp. 1837–1857.

[114] P.C. Young. "Nonstationary Time Series Analysis and Forecasting". In: *Progress in Environmental Science* (1999).

[115] P.C. Young. "Process Parameter Estimation". In: *Control and Automation Progress* 12 (1968), pp. 931–937.

[116] P.C. Young. "State-Dependent Parameter (SDP) Estimation". In: *Recursive Estimation and Time-Series Analysis* (2011).

[117] P.C. Young. "The Refined Instrumental Variable Method: Unified Estimation of Discrete and Continuous Time Transfer Function Models". In: *Journal Europeen des Systemes Automatisees* 42 (2008), pp. 149–179.

[118] P.C. Young and H. Garnier. “An Optimal Instrumental Variable Approach for Identifying Hybrid Continuous-Time Box-Jenkins Models”. In: *Proceedings of the 14th IFAC Symposium on Identification and System Parameter Estimation* 39 (2006), pp. 225–230.

[119] P.C. Young and A.J. Jakeman. “Refined Instrumental Variable Methods of Time Series Analysis”. In: *International Journal of Control* 31 (1980), pp. 741–764.

[120] P.C. Young and S. Parkinson. “Simplicity out of Complexity”. In: *Environmental Foresight and Models* (2002).

[121] P.C. Young and M. Ratto. “A Unified Approach to Environmental Systems Modeling”. In: *Stochastic Environmental Research and Risk Assessment* 23 (2009), pp. 1037–1057.

[122] W. Zhou and M. Blanke. “Identification of a Class of Nonlinear State-Space Models using RPE Techniques”. In: *IEEE Transactions on Automatic Control* 34 (1989), pp. 312–316.