Inertial and Visual Navigation Systems for Autonomous Vehicles

Dipam Chakraborty

Department of Electronics and Communication Engineering
National Institute of Technology Rourkela

Inertial and Visual Navigation Systems for Autonomous Vehicles

Thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Technology

in

Electronics and Communication Engineering
(*Specialization*: *Communication and Networks*)

by

Dipam Chakraborty

(Roll Number: 713EC5065)

based on research carried out

under the supervision of

Prof. Siddharth Deshmukh

and

Mr. Srajudheen Makkadayil

May, 2018

Department of Electronics and Communication Engineering
National Institute of Technology Rourkela

Department of Electronics and Communication Engineering
National Institute of Technology Rourkela

May 25, 2018

Certificate of Examination

Roll Number: *713EC5065*
Name: *Dipam Chakraborty*
Title of Thesis: *Inertial and Visual Navigation Systems for Autonomous Vehicles*

We the below signed, after checking the thesis mentioned above and the official record book (s) of the student, hereby state our approval of the thesis submitted in partial fulfillment of the requirements of the degree of *Master of Technology* in *Electronics and Communication Engineering* at *National Institute of Technology Rourkela*. We are satisfied with the volume, quality, correctness, and originality of the work.

Srajudheen Makkadayil
Co-Supervisor

Siddharth Deshmukh
Principal Supervisor

Department of Electronics and Communication Engineering
National Institute of Technology Rourkela

Prof. Siddharth Deshmukh
Assistant Professor

Mr. Srajudheen Makkadayil
Lead Engineer, Intel India

May 25, 2018

Supervisors' Certificate

This is to certify that the work presented in the thesis entitled *Inertial and Visual Navigation Systems for Autonomous Vehicles* submitted by *Dipam Chakraborty*, Roll Number 713EC5065, is a record of original research carried out by him under our supervision and guidance in partial fulfillment of the requirements of the degree of *Master of Technology* in *Electronics and Communication Engineering*. Neither this thesis nor any part of it has been submitted earlier for any degree or diploma to any institute or university in India or abroad.

Srajudheen Makkadayil	Siddharth Deshmukh
Lead Engineer, Intel India	Assistant Professor

Dedication

Dedicated to my parents, I would never have had the freedom to pursue my dreams without your unwavering support.

Signature

Declaration of Originality

I, *Dipam Chakraborty*, Roll Number *713EC5065* hereby declare that this thesis entitled *Inertial and Visual Navigation Systems for Autonomous Vehicles* presents my original work carried out as a postgraduate student of NIT Rourkela and, to the best of my knowledge, contains no material previously published or written by another person, nor any material presented by me for the award of any degree or diploma of NIT Rourkela or any other institution. Any contribution made to this research by others, with whom I have worked at NIT Rourkela or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the sections "Reference" or "Bibliography". I have also submitted my original research records to the scrutiny committee for evaluation of my dissertation.

I am fully aware that in case of any non-compliance detected in future, the Senate of NIT Rourkela may withdraw the degree awarded to me on the basis of the present dissertation.

May 25, 2018
NIT Rourkela

Dipam Chakraborty

Acknowledgment

May 25, 2018
NIT Rourkela

Dipam Chakraborty
Roll Number: 713EC5065

Abstract

Indoor navigation is a challenging task due to the absence of Global Positioning System(GPS). This project removes the need for GPS in systems by combining Inertial Navigation Systems (INS) and Visual Navigation Systems (VNS), with the help of machine learning with Artificial and Convolutional Neural Networks.In GPS denied environments a highly accurate INS is necessary, it must also be coupled with another system to bound the continious drift error that is present in INS, for which VNS is employed.

The system was implemented using a ground robot to collect ground truth data, which were used as datasets to train a filter that increases the accuracy of the INS. The accuracy of the INS has been proven on the hardware platfrom over multiple datasets. Eventually Visual Navigation data can also be fed into the same system, which for now is implemented in simulation, as an independent system. A software and hardware framework have been developed that can be used in the future for further developments. The project also optimizes visual navigation for use on low power hardware with hardware acceleration for maximized speed. A low cost and scalable indoor navigation system is developed for indoor navigation, which can also be further extended to Autonomous Underwater Vehicles (AUV) in 3D space.

Keywords*: *Navigation Systems*; *Inertial Measurement Unit*; *Visual Navigation*; *Machine Learning*; *Convolutional Neural Network*; *Hardware Acceleration

Contents

List of Figures

Chapter 1

Introduction

Navigation Systems provide velocity, orientation and position of a body. These are essential for localization of an autonomous robot/vehicle, as well as the control systems. Current systems make use of combinations of GPS, External Cameras, and Laser Rangefinders along with Inertial Navigation Systems. However, these systems are not always available in all scenarios and environments.

For GPS denied environments, such as indoors, mines, and underwater environments, accurate navigation systems are difficult to develop. Especially on-board navigation systems in unstructured environments are the ultimate target in the field. Many strides have been made in developing solutions to particular environments. For example, in underwater systems, Doppler Velocity Logs (DVL) provide accurate velocity measurements and are currently the state of the art in the field, SONAR systems have are also been used for larger submarines and ships. For ground and aerial vehicles, Light Detection and Ranging (LiDaR) and Radio Detection and Ranging (RaDaR) have also been developed. Off-board solutions based on Multiple cameras and markers are used in closed environmets to simulate GPS like navigation.

1.1 Types of Navigation Systems

Desouza et al. mention in their survey of robot navigation systems [1] about the different types of navigation systems. The figure 1.1 illustrates the various kinds of navigation systems. Small robots working in indoor environments can work on structured or unstructured navigation systems based on the system requirements. Often it is the case that highly accurate navigation is not required by the problem statement, and the algorithms are robust to inaccuracies in short periods of time. Inertial Navigation systems [2] use inertial sensors and continious integration to estimate the navigation paramers. All navigation systems typically are augmented with other ones, requiring sensor fusion of some sort, various versions of the kalman filter [3] are widely used for these kind of applications.

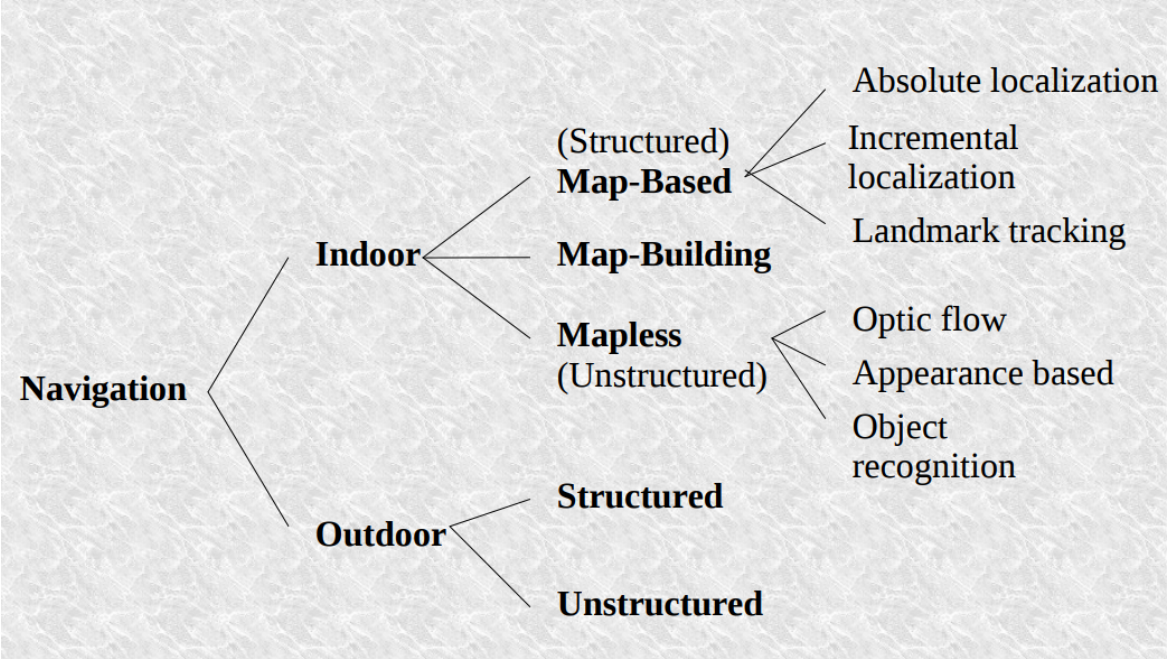

Figure 1.1: Different kinds of Navigation Systems

1.2 Related Work

Many researchers have previously tried many different approaches to improve accuracy of INS augmented systems. Below some of these methods are discussed, some of these methods use GPS, but it is useful to understand how GPS/INS integration has been done to implement it ourselves.

- Won et al. have explored the idea of combining INS with Visual SLAM in their paper [4]. They use a distributed particle filter to address the visual slam, along with INS to predict sub states in their slam algorithm to limit the number of features that are to be tracked. This reduces the computational requirement and hence allows higher accuracy by faster sampling of data.

- This work[5] on hybrid fusion uses two IMUs in an environment with massive GPS outage probablity, hence requiring an accurate method to track without GPS for a long period of time. They do this by combining two method, first to combine using a tightly coupled physical setup, and the second using the fact that most pedestrians take one step at a time, so this pattern of movement is tracked.

- Bhatt et al. also uses a machine approach to the INS/GPS integration problem.[6] They used Support Vector Machines (SVMs) to augment the Dempster–Shafer fusion methodology.They state that during outages, the SVM is successfull in correcting the INS sufficently during the outage period. They argue SVMs perform well in this regard and also ANNs have scope for using a similar approach.

- Ning et al. discusses the application and development of such a system for planetary rovers, for deep space exploration [7]. They use a three-fold combination of inertial, visual and celestial navigation systems to produce accurate results in unstructured simulation environments. For sensor fusion and state estimation, they employ the unscented kalman filter(UKF). [8]

- Designing systems that can run on low power is essential for scalability. This is captured in this paper [9], where they have designed a probabilistic framework for running on a low power cpu with only one core. They use a streo capture camera and an IMU for odometry estimation and tested their navigation system on a multirotor aerial vehilce and a quadreped robot.

- Sebesta et al. uses the Adaptive high-gain extended kalman filter (AEKF) [10] on fast moving umanned aerial vehicles [11]. AEKF has been proven to converge globally. They argue AEKF is a robust approach to combine noisy INS data.

- Underwater sensors have so been used with INS systems, Tal et al. have wokred on combining DVL and INS data[12] in their paper. They use IMU as well as pressure sensor for depth measurement, their assumption is to use DVL as a loosely coupled device and simulate the accuracy improvement by using this method.

1.3 Motivation

The above mentioned solutions face a few disadvantages, DVLs and SONAR systems are very expensive, power inefficient, and bulky when considering small Autonomous Underwater Vehicles (AUV). For comparision, a fully equipped AUV can be designed within a weight margin of 20 kgs [13] [14]. Hence it become necessary to have a navigation system for such vehicles. This kind of a system can also be used in indoor systems where GPS is not available.

1.3.1 Advantages

The advantages of such a system are:

- Low cost and scalable
- Draws less power, hence good for long endurance
- An order of magniture reduction in weight
- Works in multiple environments

1.3.2 Challenges

IMU measurements can be reliably used to measure the heading and attitude angles of a robot. However, there are many challenges faced when using only inertial sensors for navigation.

• Small errors in acceleration can add up to make the position drift very fast.

• Gravity cancellation requires perfect orientation measurement, while it is quite accurate, no measurement is perfect and even small errors can increase the error in the gravity cancellation, having a constant dc bias in the acceleration.

• Vibrations can cause significant noise in inertial sensors

• While the system is not moving, the position can still drift away because of residual velocity measurments

• The nature of movement of an Autonomous System is difficult to fully model in a real-world scenario.

1.4 Objective

The project aims to train an adaptive filter with the help of a more reliable velocity measurement sensor such as a wheel encoder, to form a training dataset so that the difference in the acceleration can be compensated. An actuator control feedback model is employed to reduce the bias term when the robot is not expected to move. MEMS based inertial sensors and Cameras provide a low power, cost effective and light weight solution. This project aims to develop methods to use these sensors machine learning and computer vision for sensor fusion and filtering.

1.4.1 Methodology

This project aims to solve some of these problems in the following ways

- Collect datasets on a ground vehicle mounted with one or more IMUs, and with wheel encoders as a ground truth reference.
- Machine learning will be used on this collected data train a neural network to filter the sensor data.
- After training, the IMUs can be mounted as standalone units in robots which do not have wheel encoders available to them.
- To address the problem of no-movement, actuator feedback will be taken (whether the actuators are running and the targeted speed), to get a force estimate on the robot, for better INS estimation.

It shall also make the use of computer vision in the latter part of the project to improve the accuracy by integrating cameras along with the inertial navigation system. These algorithms can use Visual Simultaneous Localization and Mapping (V-SLAM) [15] which employs generalized feature matching using neural networks, making them robust for use in any scenario both known and unknown.

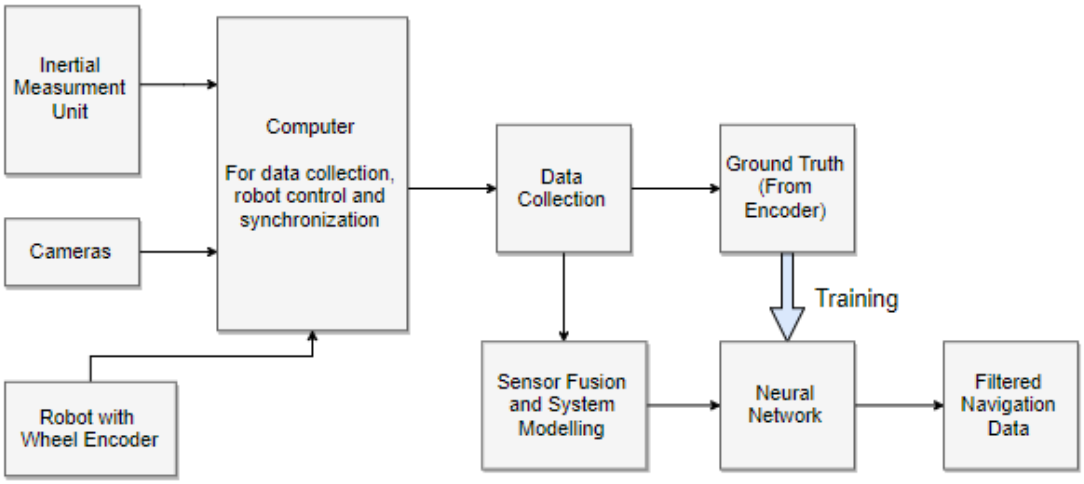

Figure 1.2: Functional Block Diagram of the project

The functions of individual blocks are explained below in brief:

- **Inertial Measurement Unit (IMU) :** For sensing accelerations, angular velocities and magnetic field strength.
- **Cameras :** For detecting objects and estimating the robot's position using vision.
- **Wheel Encoder :** For counting wheel revolutions for as a ground truth reference.
- **Data Collection :** Forming datasets for testing different offline training algorithms.
- **Synchronizer :** For synchronous data collection by interfacing all the different sensors (Multiple IMUs, Cameras, Wheel Encoders).
- **Sensor Fusion :** Applying necessary rotations, time shifts, normalizations and transforms necessary for making all the bringing all the sensor data into a single reference frame.
- **Neural Network :** To apply various machine learning and deep learning algorithms in order to filter and fuse the different sensor data.
- **Ground Truth :** – A reference system made from encoder data to train the neural network.

Chapter 2

Fundamentals of Inertial and Visual Navigation Systems

This chapter goes through the basic theory of both Inertial Navigation and Vision based Navigation systems. It also discusses the sensors and method used to collect the ground truth to train and test the navigation data. Modifications made to the INS algorithm are also discussed.

2.1 Inertial Navigation

Inertial Navigation System (INS) uses dead reckoning (continuous integration) on linear accelerations, estimate the velocity and position, based on a known initial velocity and position. It fuses data from sensors, mainly inertial measurement units (IMUs), more than one IMU may be used for better estimates, robot movement model and other navigation systems like GPS can be integrated with INS.

2.1.1 Inertial Measurement Unit

An Inertial Measurement Unit is a combination of Accelerometers, Gyroscopes and Magnetometers. In all 3 axes of rotation and translation, which combined gives a total of 9 Degrees of Freedom for the sensor[2]. Accelerometers measure linear accelerations, which can be used to find the direction of the earth's gravity. Gyroscopes measure angular velocities, which are used to estimate changes in a rotating body's orientation. Magnetometers measure the strength of the earth's magnetic field in each axis, giving the direction of the true magnetic north relative to the sensor. Figure 2.1 depicts a body in a body fixed frame, most sensor measure data in this frame.

IMUs based on Micro Electro-Mechanical Systems (MEMS) are the cheapest and suitable for smaller robots [16]. Most MEMS IMUs come equipped with internal microcontrollers to combine the sensor data to give accurate Orientation data. This uses an Attitude Heading Reference System (AHRS) [17] Kalman filter. The accuracy in general depends on the grade of MEMS technology used. The accuracy is within 1 degree of error and is good enough for our application.

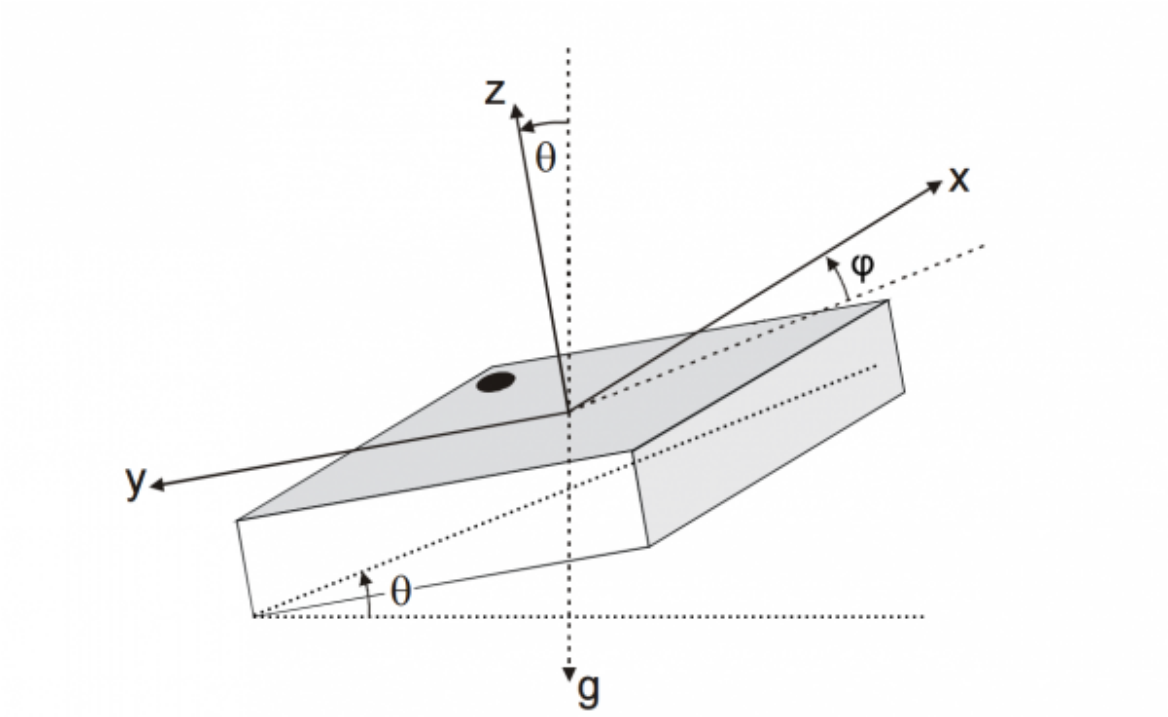

Figure 2.1: Inertial Measurement Unit in body fixed frame

The accelerometers provide 3 Axis measurements of combined Linear Acceleration and Gravity. To find the true body acceleration of the module, the gravity component has to be estimated and removed, this is generally prone to noise and a non – zero bias that may vary with time, especially when the sensor is moving.

MEMS accelerometers also suffer from additional noises due to sampling, capacitive slowdown, and imperfect A/D conversion[18]. They also need to be calibrated, and accuracy depends on the technology used to manufacture them, so cost and accuracy of these sensors can vary widely.

2.1.2 Inertial Navigation Algorithm

Euler rotations: Euler Angles represent a body's attitude as a tuple of three angles Yaw (ψ), Pitch (ϕ), Roll (θ). [2] This can have with respect to a body fixed frame or an inertial frame, which is set according to the earth's gravity and the robot's starting pose. To convert from one frame to another using Euler Angles, the Direction Cosine Matrix (DCM) is multiplied with the acceleration vector.

Euler angles are used for their simplicty in representing the orientation of a body in 3D space, other methods like quarternions can also be used, but Euler Angles are the most widely used in INS implementations.

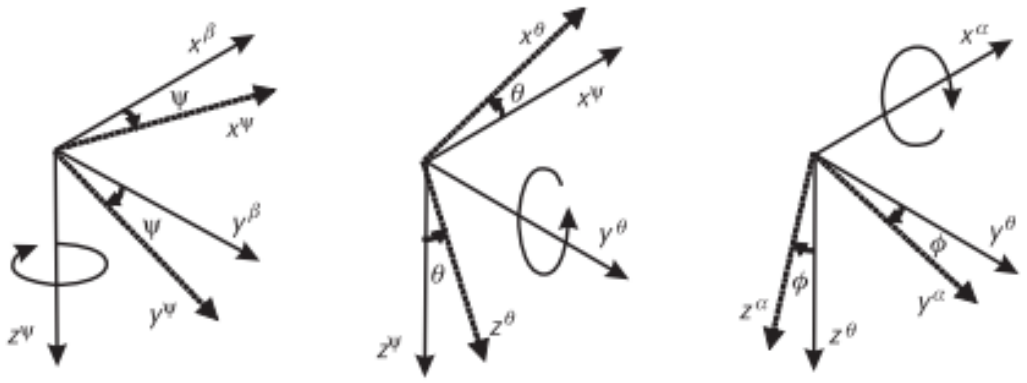

Figure 2.2: Euler rotations in YPR order

$$C_{\alpha}^{\beta} = \begin{bmatrix} \cos\theta_{\beta\alpha}\cos\psi_{\beta\alpha} & \begin{pmatrix} -\cos\phi_{\beta\alpha}\sin\psi_{\beta\alpha} \\ +\sin\phi_{\beta\alpha}\sin\theta_{\beta\alpha}\cos\psi_{\beta\alpha} \end{pmatrix} & \begin{pmatrix} \sin\phi_{\beta\alpha}\sin\psi_{\beta\alpha} \\ +\cos\phi_{\beta\alpha}\sin\theta_{\beta\alpha}\cos\psi_{\beta\alpha} \end{pmatrix} \\ \cos\theta_{\beta\alpha}\sin\psi_{\beta\alpha} & \begin{pmatrix} \cos\phi_{\beta\alpha}\cos\psi_{\beta\alpha} \\ +\sin\phi_{\beta\alpha}\sin\theta_{\beta\alpha}\sin\psi_{\beta\alpha} \end{pmatrix} & \begin{pmatrix} -\sin\phi_{\beta\alpha}\cos\psi_{\beta\alpha} \\ +\cos\phi_{\beta\alpha}\sin\theta_{\beta\alpha}\sin\psi_{\beta\alpha} \end{pmatrix} \\ -\sin\theta_{\beta\alpha} & \sin\phi_{\beta\alpha}\cos\theta_{\beta\alpha} & \cos\phi_{\beta\alpha}\cos\theta_{\beta\alpha} \end{bmatrix}$$

Figure 2.3: Direction Cosine Matrix

$$x_{\delta\gamma}^{\beta} = C_{\alpha}^{\beta} x_{\delta\gamma}^{\alpha}$$

Here x may represent any vector in 3-dimensional space, accelerations, velocity and positions. These may be rotated back by the transpose of the DCM matrix. Consecutive rotations can be modelled as a multiplication of DCM matrices in the same order of rotation.

Gravity Cancellation: The accelerometer measures a combination of Earth's gravity and the linear acceleration due to movement. Thus, to find the true body acceleration of a body, the gravitational components have to be removed from the sensor values. To do this, the sensor gravity vector is rotated to the body frame and then subtracted. As can be seen from equation 2.1, after finding the true body acceleration, the inverse DCM is then used to convert back to the Inertial Frame.

$$\begin{aligned} a_b &= a_m - C_I^B * (00g)^T \\ a_i &= C_B^I a_b T \end{aligned} \tag{2.1}$$

Dead Reckoning: Linear accelerations as measured from the sensor are integrated doubly over time to find the position estimate. This estimate has a high degree of error for an uncalibrated sensor and hence has to be calibrated to reduce the error. Furthermore, the situation of no movement poses a challenge since non-zero acceleration bias results in position drift even when the robot is not moving. Equations 2.2 shows the dead reckoning eqautions.

$$\begin{aligned} v_i &= \int a_i \leftrightarrow \Delta v_i[n] = a_i[n]\Delta T \\ r_i &= \int\int a_i \leftrightarrow \Delta r_i[n] = v_i[n]\Delta T \end{aligned} \tag{2.2}$$

2.1.3 Problems of INS

The independent INS algorithm is not very reliable at low vehicle speeds, as there can be many problems[19], as listed below

- This estimate has a high degree of error for an uncalibrated sensor and hence has to be calibrated to reduce the error.
- The situation of no-movement poses the problem that a non-zero, time varying bias in the acceleration data is integrated over time and the error keeps adding up.
- During movement, gravity cancellation can be inaccurate due to vibrations, which makes small variations in the acceleration data can eventually lead to being added up overt time.
- MEMs can accurately measure large values of accelerations, however when the acceleration is close to 0, the errors can be very significant.

2.2 Ground Truth

For ground truth, wheel encoders, and highly orientation data IMUs have been used. Some implementation ideas were adopted from [20], details are given below

2.2.1 Sensors for ground truth collection

Wheel Encoder: Ground Vehicles mounted with motors can use wheel encoders to measure their velocities. A wheel encoder has a "spoke wheel" and an Infrared Rx/Tx Pair. The spokes are marked so that they block the light from an IR Tx from reaching the other side, where a IR Rx is placed. The receiver is active high, so it produces when light is IR light incident on it, and low otherwise This causes a series of pulses when the wheel is moving.

The time difference between two rising edges of the sensor data can be measured, since distance between two of the spokes is fixed, this is used to calculate the velocity. Figure **??** shows a detailed diagram of a wheel encoder.

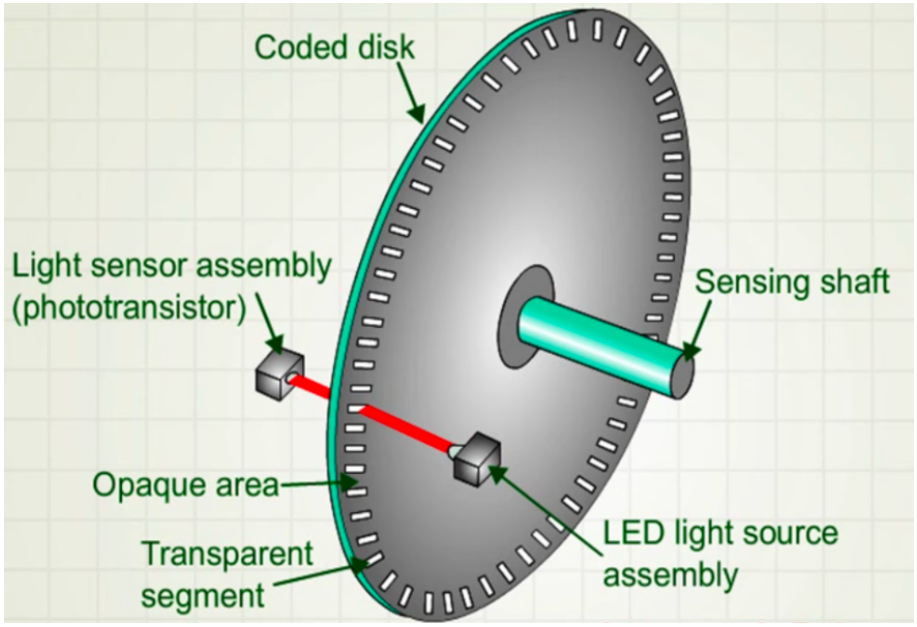

Figure 2.4: Wheel Encoder

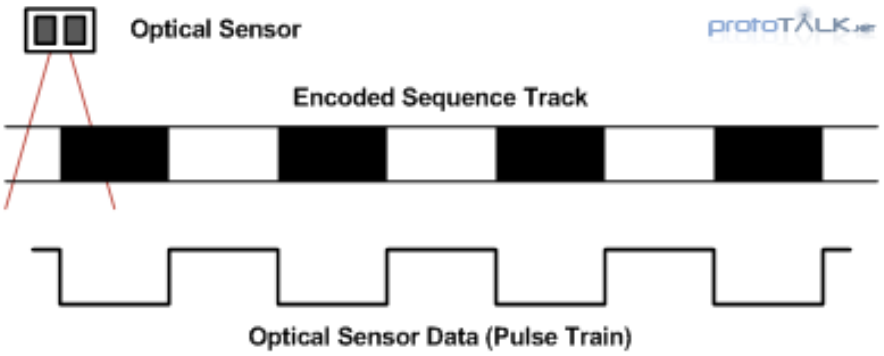

Figure 2.5: Encoder Signal Visualization

Speed Measurement by encoders: Encoder speed measurement uses the time difference between two rising edges to compute the velocity. This requires prerequisite knowledge of the wheel diameter, vehicle dimensions and number of ticks per revolution for the encoder. Equations 2.3 are used to calculate the speed.

$$
\begin{aligned}
v_w &= \frac{\pi D_w}{T_{edge} * Ticks_{rev}} \\
v_f &= \frac{v_r + v_l}{2} \\
w &= \frac{v_r}{R} = \frac{v_l}{R + l_w}
\end{aligned}
\tag{2.3}
$$

2.2.2 Algorithm for ground truth collection

Ground Truth is obtained from the most accurate information available. There will be small error in ground truth due to measurement error, but it is the best available estimate of how the robot moved, hence will be used the baseline. Ground truth equations 2.4 are used to compute the ground truth. Where θ is taken from IMU orientation data.

Ground truth uses the following sensor data:

- Encoder Velocity Data
- IMU Orientation Data
- Overhead Camera based tracking (not implemented in this project)

$$
\begin{aligned}
v_{forward} &= \frac{v_{right} + v_{left}}{2} \\
v_{inertial} &= (cos\theta, sin\ \theta) * v_{forward} \\
r_i &= r_{i-1} + \frac{v_{inertial}}{freq_{sample}}
\end{aligned}
\tag{2.4}
$$

2.3 INS Modifications

For higher accuracy in the INS algorithm, some modifications were made.

2.3.1 Dual IMU Fusion

More than one IMU can be used for more accuracy[6] [21]. Figure 3.6 shows a block digram of the process. Two IMUs were mounted on the robot. Initially, an inertial frame was fixed. Since both IMUs measure in different body fixed frames, both are converted to the inertial frame. Then both of them are again converted to the robot's body fixed frame, this ensures high data consitensy in one of the dimensions which is primarily the direction in which the robot moves forward.

2.3.2 Robot Control Feedback

To reduce the problem of non-zero acceleration bias, an actuator feedback model has been implemented. This model reduces the acceleration measurement bias sharply to zero as the

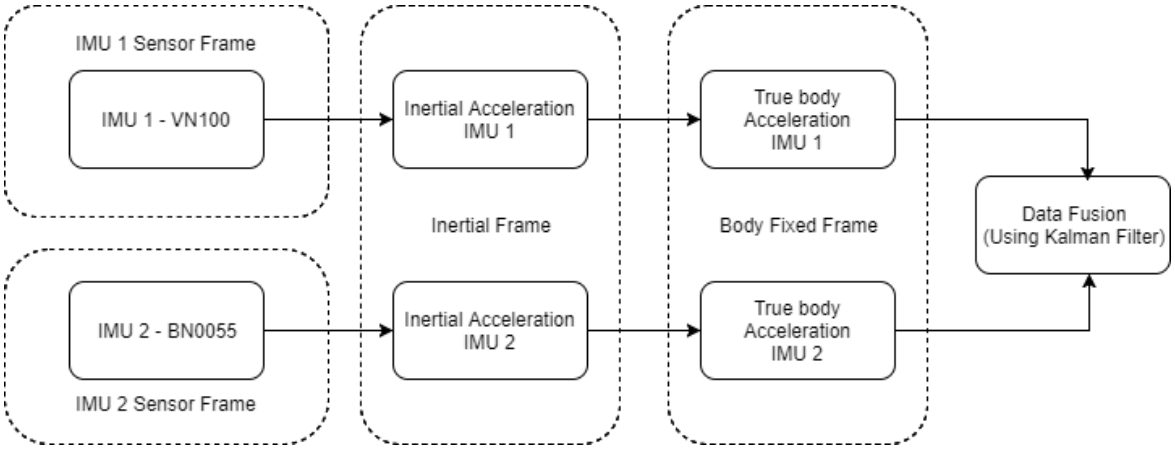

Figure 2.6: Dual IMU data fusion

robot stops its motors. The measurements are hence taken only when the robot is expected to move, greatly reducing the drift robot's position drift caused by dead reckoning.

Movement can be estimated based on control information sent to robot's actuators. In this case, it is PWM value sent to wheels. Sent PWM data is differentiated to find change in speed with the assumption that wheel RPM is proportional to pwm within practical speed range. The differentiated PWM data is convolved with a window function from equation 2.5 to create a acceleration filtering window. The function falls towards 0 as x gets greater than a, the steepness of descent is decided by the other constant b. Control data is used to estimate the following:

- Whether the robot is expected to be moving.
- Whether the robot is expected to change speed.
- Estimated acceleration if speed is changed.

$$f(x;a,b) = \frac{1}{1+|\frac{x}{a}|^{2b}}; x > 0 \tag{2.5}$$

2.3.3 Filtering using Neural Networks

Machine learning is used to automate modelling of a system by using data and without being explicitly programmed to find particular features. This may use many models such as Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Genetic Algorithms, etc. [22]

Artificial Neural networks are inspired from biological structures of neurons, which get tuned adaptively to data filtering, prediction or classification. These convert lower dimensional data into combinations of higher dimensional data, more number of hidden layers allows the network to train on features and combinations of features.

For INS, the past $freq_{sample}/2$ number of points were passed into a neural network for training. The first layer had 50 neurons, the second layer had 25 and the final layer was a

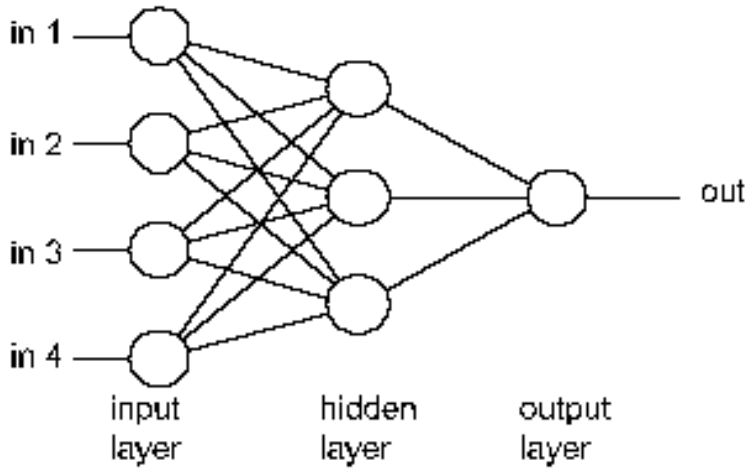

Figure 2.7: Depiction of a small Artificial Neural Network

single output neuron. The training was done on velocity data obtained from dead reckoning on acceleration. Activation function was hyperbolic tangen all throughout the network. All connections were fully connected. Backpropogation with stochastic gradient descent (SGD) was used for training on 20 datasets with 0.5 dropout on 10 epochs, batch normalization size of $freq_{sample}$ was used. Loss function was L2 regularized mean square error (MSE).

2.4 Vision based Navigation

Vision based navigation relies on cameras as the main sensor for inferring navigation variables[1]. One or more than one camera may be used for this kind of navigation system. For indoor navigation, according to figure 1.1, the options range from mapless, map-building and map-based navigation schemes. Mapless navigation can be especially challenging due to changes in point of view, features in the environment and other factors. Map-buidling applications like Visual Simultanoeus Localization and Mapping (V-SLAM) assumes most of the environment is constant and sufficiently distinguishable from other locations. This requires more computation, but is more robust. Map-based methods especially landmark tracking is one of the simpler approaches, this project shall incorporate this basic system and implement it with hardware acceleration, as well as some degree of robustness.

2.4.1 Convolutional neural networks

Cinvolutional neural netrworks are a form of Artificial neural networks that uses weight and connection sharing to massively reduce the amount of computations required on large images. CNNs are primarily used to solve difficult image-driven pattern recognition tasks and with their precise yet simple architecture. [23]. Figure 2.8 shows the architecture of a typical convolutional neural network, the number of layers and other hyperparameters may be altered. Many different architechtures of CNNs have been developed for different tasks.

[24]

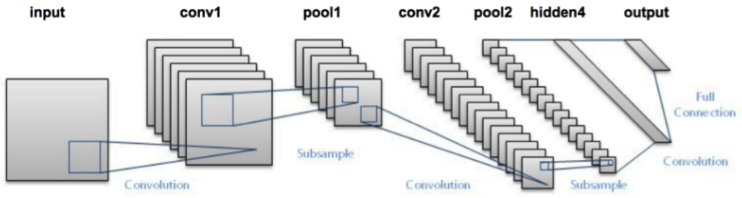

Figure 2.8: Architecture of Convolutional Neural Network

2.4.2 Object Tracking

Landmark tracking based apporaches, as the name suggests, requires tracking of known objects in the environment. There are many architechutues for this purpose[25] [26] [27], but a recent convolutional neural network system combines object classification, and localization into a single regression problem has been recently developed. The 'You Only Look Once' (YOLO) [28] is a state of the art object tracking CNN architecture that has a appreciably low computation time. This has been tested on machines equipped with modern GPUs. This work further reduces its computation requirements by limiting its problem statement to fewer labels - 30 instead of 1000, and accelerate it on FPGA hardware. Figure 2.9 shows an example of the original YOLO network tracking objects in a given image.

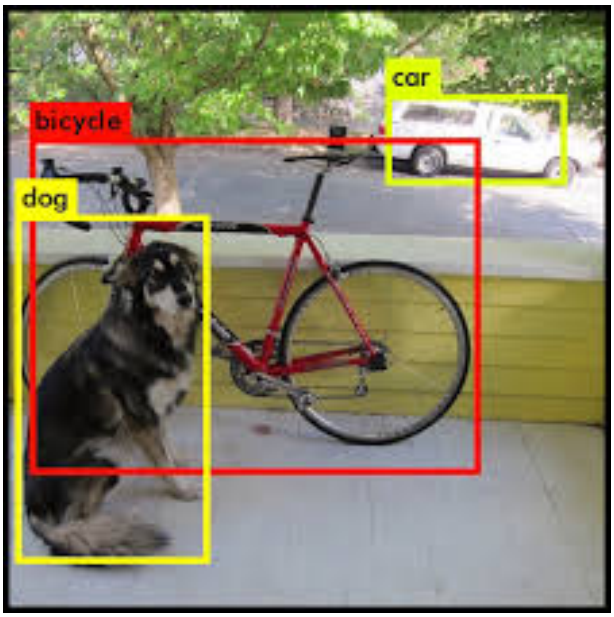

Figure 2.9: Output of the original YOLO network

2.4.3 Map based tracking

The landmark tracking approach requires us to have a known map of the environment. [29] The landmarks may have different levels of features made available to us. The spatial

coodinates of the landmarks, with the assumption that they are stationary. See figure 2.10 for an abstract example. The star marked objects are the depicted landmarks, they should all be unique. The approximate size of each landmark is also required in order to estimate distances.

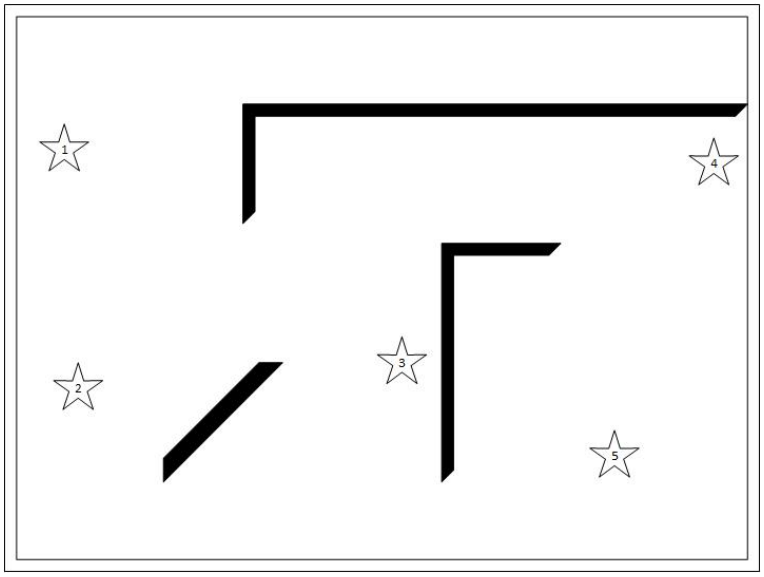

Figure 2.10: Abstract depiction of a map

2.4.4 Visual Geometry

Visual geometry is used to convert from the image coordinates, obtained from object tracking, to real world coordinates. Here it is used to find a distance and angle estimate relative to the object that is being tracked [30].

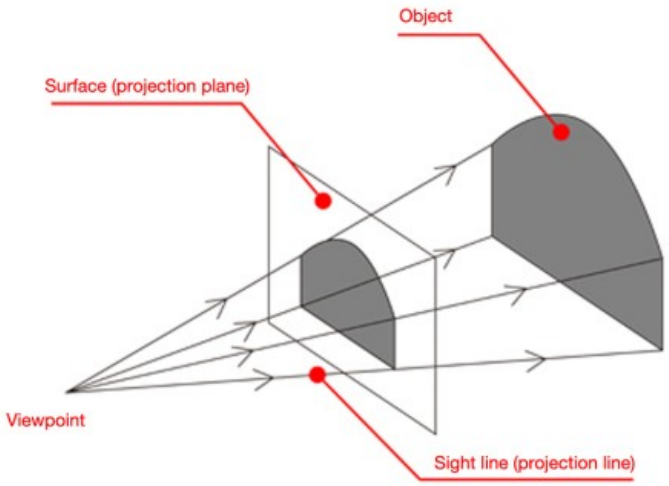

Figure 2.11: Projection of a body to the image plane

Any real world object projects itself onto the image plane, which is also called the

projection plance. The size of the object is scaled based on the image size and distance of the object. As seen from figure 2.11, the object and projected image, form similar triangles when connected by the projection lines. In this triangles, two parameters are known. The distance between the veiwpoint and the image plane [31] , and the size of the object in the image plane (in pixels). The unknown variables in this are the size of the object in the real world, and the distance of the object. Hence, if size of the object can be known, the distance can be estimated from the similar triangles, as shown in figure 2.12

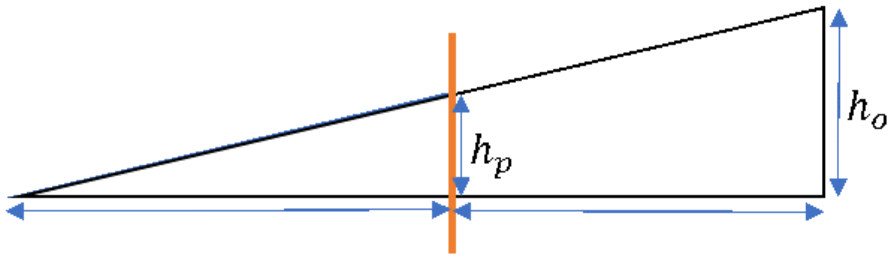

Figure 2.12: Similar triangles from perspective projection

Distance d_o can be calculated from

$$\frac{d_p}{d_p + d_o} = \frac{h_p}{h_o}$$

Where h_p can be calculated from the sensor height, by scaling it by the number of rows in the image.

2.4.5 Visual Navigation Algorithm

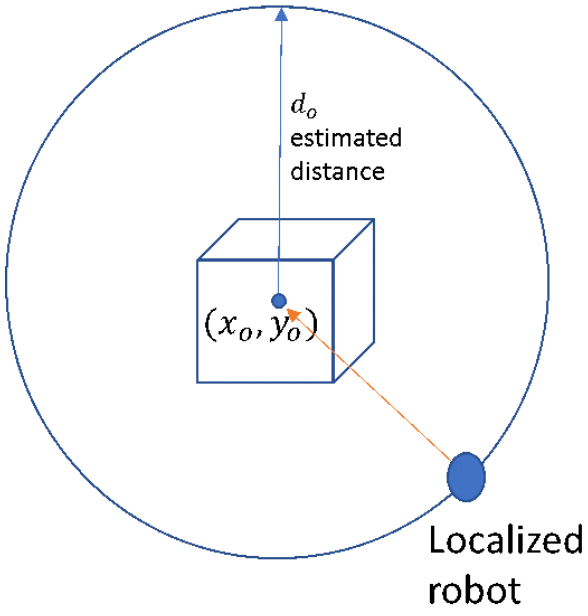

Figure 2.13: Localization from distance and angle to a known landmark

If a known map object is at (x_o, y_o) with estimated distance d_o. The robot will be on the circle depicted. The angle from which the robot is viewing is required to localize. The required values are Robot heading angle (from IMU), Camera angle (for rotating camera) and object angle in projection plane with respect to the center. Combining all three angles gives the angle normal to the circle of localization.

2.5 Sensor Fusion

The kalman filter [3] has long been regarded as the optimal solution to many tracking and data prediction tasks. It is widely used in navigation systems for fusing data and model predictions of different accuracies. It works like a weighted mean of two different normally distributed data after both are transformed into the same observation space. The result being more precise than each of them individually.

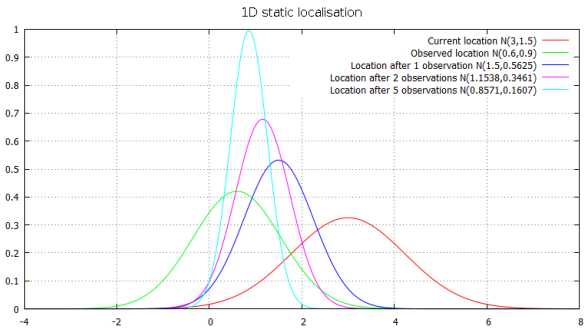

Figure 2.14: Visualization of a Kalman filter with multiple data

The figure 2.14 shows the accuracy of a kalman filter increasing by combining more datapoints. The Kalman filter is used even in GPS/INS systems [32] and hence this project is motivated to use them for effective VNS/INS integration.

Chapter 3

Implementation and Results

This chapter discusses the implementation details of the project, and the results obtained. The results are presented in chronological order as new ideas and modifications were tested.

3.1 INS Implementation

The INS system was implemented on a ground robot, mounted with 2 IMUs, wheel encoders, and interfaced with Arduino and Robot Operating System (ROS) on a host machine for data collection and processing.

3.1.1 Overview

INS was integrated as three sub-parts, depicted in block diagram in figure 3.1 the first, the main VN100 IMU sensor unit which has an inbuilt processor and factory calibration settings, these were simple API calls without any algorithmic implementation. It is recommended use the same as the most accurate model is integrated into the IMUs own system and is proprietary to the manufacturer. Second is the ground robot that was used to collect the data on, it also has the wheel encoders which provide the ground truth. Lastly the computer that acts as the synchonizer between all sensors for data collection, and also as the offline processing unit.

The VN-100 IMU module has the AHRS Kalman Filter and Gravity Cancellation algorithms implemented already, they have to called using the sensor's library functions. The INS algorithm and necessary actuator feedback based transforms are implemented on the computer and collected into datasets for offline training. The ground vehicle receives the higher-level control commands from the computer and interfaces the wheel encoder. It also provides the actuator feedback according to the current state of the actuators being driven.

The neural network training is done offline on the datasets taken from the robot. All the relevant data is collected initially but passed through normalization and rotations in order as pre-processing before passing through the neural network. Since this uses temporal information, recurrent neural networks are used for this purpose.

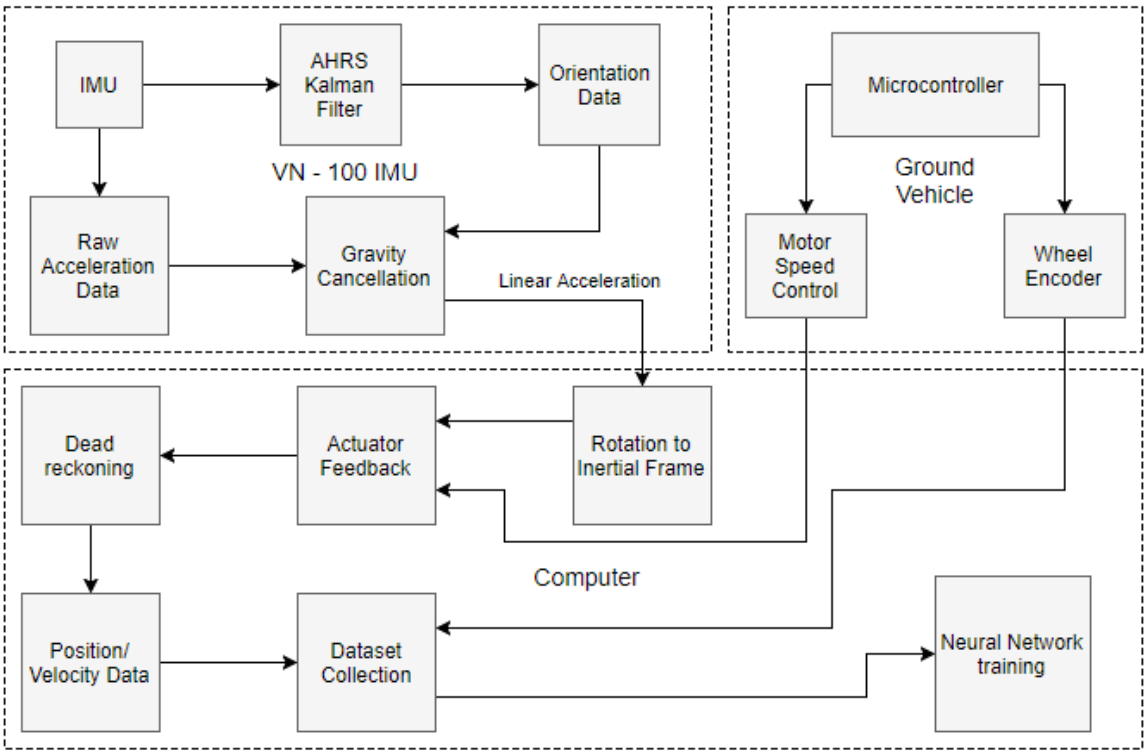

Figure 3.1: Detailed block diagram of INS

3.1.2 INS Hardware

The ground robot shown in the photograph of figure 3.2 was used to collect IMU data and ground truth, the details of the robot's hardware are given below:

- **IMU:** VectorNav VN100. An IMU that can be interfaced with a computer over USB. It has 9 DOF and an internally compensated magnetic data matrix for accurate angular measurement. BN5505, another IMU but less expensive and less accurate than VN100, interfaced with microcontollier over I2C.
- **Drive Mechanism:** The robot uses differential drive mechanism for movement using 2 independent motors which can rotate both forward and backward.
- **Wheel Encoder:** The individual wheel encoders used for both the wheels are rated at 32 ticks per revolution. No direction measurement is provided, it is done using actuator feedback.
- **Microcontroller:** Arduino Nano, equipped with Atmel ATMega328p running at 16 MHz. It is used to control the robot from the computer and receive wheel encoder and actuator feedback data.
- **Power:** The robot has self-contained power using a Li-Po Battery.
- **Motors:** Each individual motor is rated at 200 RPM under no load at 12V.

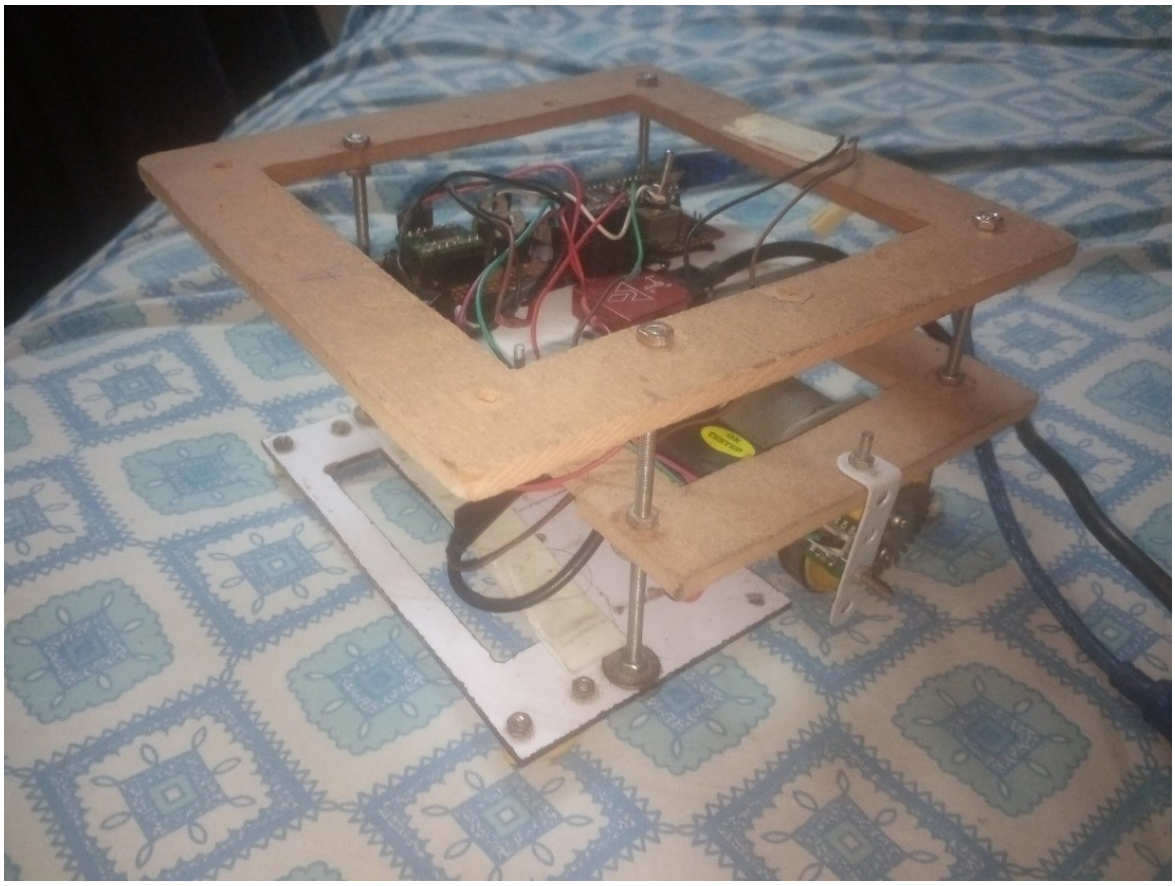

Figure 3.2: Ground Robot used for data collection

3.1.3 INS Software

Sampling Rate Tradeoff IMU provides orientation data as well as accelerations. With a higher sampling rate, more noise is introduced into these data [18]. But with a lower sampling rate, there may be loss of essential acceleration information. Orientation data is cruicial and its accuracy should not be reduced. The VN100 has an internal vector processing unit which calculates orientation, if data is sampled at higher rate, orientation data will be less accurate. Hence a tradeoff point is set at 50 Hz, where orientation errors were emperically found to be less than 1 degree. So 50 Hz was set as the sampling frequency.

Hardware Interfacing – Arduino: The flowchart in figure **??** shows how the arduino was interfaced. It does three primary tasks

- **Robot Control:** Takes user commands and controls the robot motor's speed and direction.
- **Encoder Polling:** Samples the encoders on hardware interrupts to measure the time difference.
- **Data Transmission:** Transmits data at a fixed frequency, this includes encoder velocities and robot's movement direction and pwm values.

Hardware Interfacing - VN-100 IMU:

The VN-100 IMU is initalized with the following parameters:

- Sampling Frequency - 50 Hz
- Data Mode - Asynchronous
- Data values :
    - Orientation (Yaw, Pitch, and Roll)
    - Gravity cancelled linear accelerations in body frame
- Frame initialization - Global reference frame is initialized with Tare command

After data packet is recieved, timing information is appended on each data packet.

Matlab Progams

- MATLAB node communicates with ROS
- ROS sends all information from robot and user interface
- Data Collection:
    - Robot movement state
    - Robot orientation
    - IMU Values (Accelerations and Angular Rates)
    - Encoder ticks
    - Timing information
- Offline data processing:
    - Noise Removal
    - INS Algorithm
    - Neural Network Training

3.1.4 ROS - Robot Operating System

ROS (Robot Operating System) provides libraries to help create robot applications [33]

- It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source license.
- ROS alleviates the problem of developing programs for synchronized processing in real time. Often faced when dealing with multiple sensors, actuators, low level data processing routines etc., which are very common in modern robots.
- ROS provides a system of defining modular user defined “nodes”, and manages timing, permission and data management in the background, like a scaled down operating system.

- Data is passed from one node to another through a "ROS Master" node, through a subscriber/publisher protocol implemented in ROS.
- Since ROS master is a network location, multiple computers can communicate with the same master and hence networking operations are also made simpler through ROS.
- Package management is an important feature in ROS, as this allows users around the world to develop general purpose packages, which are available as open source modules.

Currently ROS can only run stably on Linux Based Operating Systems, such as Ubuntu.

3.2 INS Results

This section goes through some characteristic plots of the different experiments and modifications done on the INS algorithm. All results shown are presented in chronological order. All acceleration plots are shown in the robot's body fixed frame in the forward direction of movement.

3.2.1 Basic results with linear filter

Figure 3.3 shows a plot of unfiltered data acceleration data in the robot's forward movement direction from the VN100 IMU. The corresponding encoder acceleration has been calculated by differentiating the forward average veloity of the ground truth data.

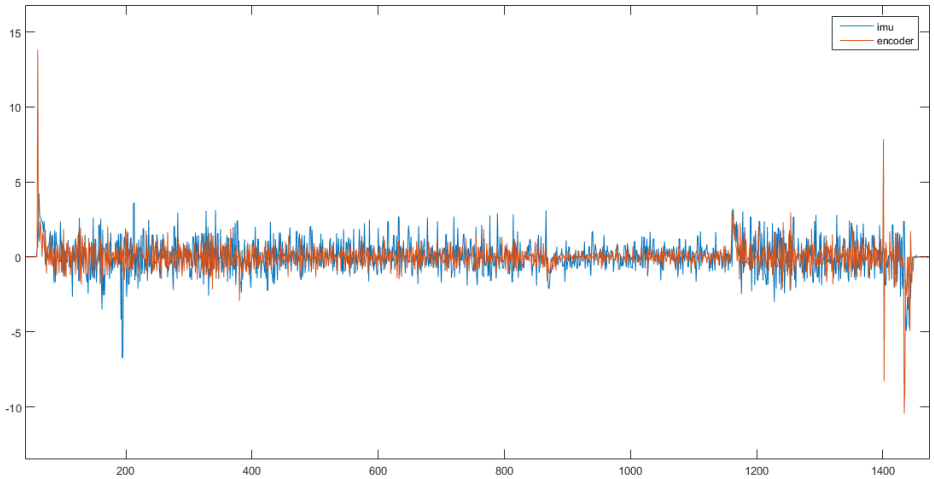

Figure 3.3: Unfiltered IMU data plot with encoder acceleration

Since it is clear that the data very noisy, a basic sliding window averaging low pass filter is applied on the data as preprocessing. The result is shown in figure 3.4. As can be seen,

the acceleration noise can be sometimes as high as a real signal, hence it can be inferred the SNR of these readings is very poor.

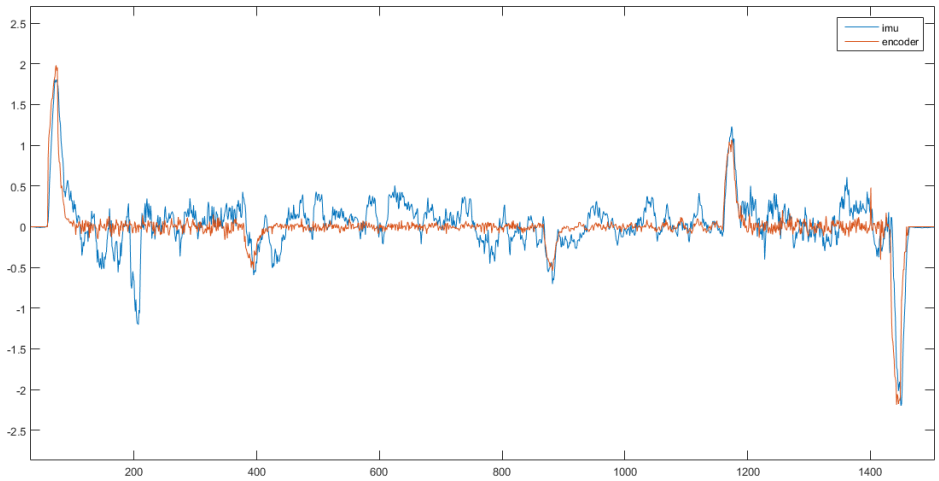

Figure 3.4: Acceleration data plot with simple sliding window averaging filter

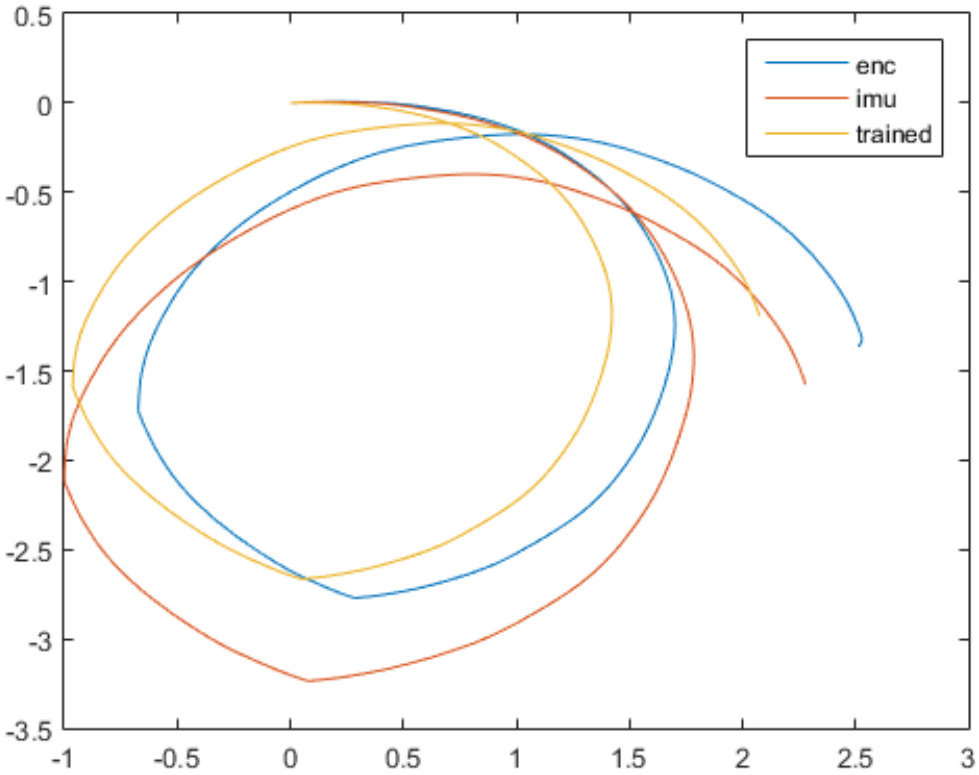

Figure 3.5: Position plot from linear filter

Upon applying a linear adaptive filter on the basic acceleration data, then applying dead

reckoning according to the INS algorithm, the result are found to be fairly poor, as is evident from figure 3.5, implying that a linear filter will not be sufficient to improve the accuracy. Hence we move to neural networks.

3.2.2 Dual IMU results

Data collected from both IMUs are used to find the variance of both, against the ground truth. The different variance values are used as a weighting factor in the Kalman filter to improve the accuracy by moving closer to the expected value. Figure 3.6 shows the data from both IMUs while figure 3.7 shows the result of the kalman filter combination.

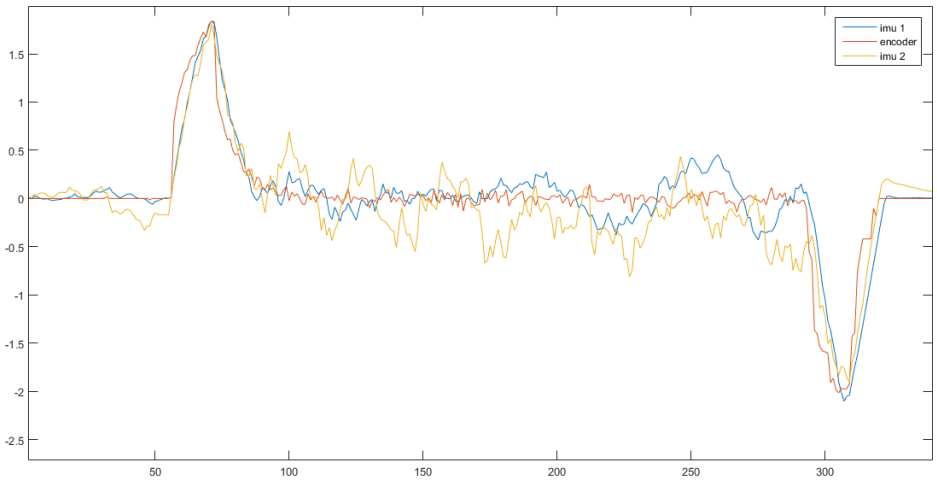

Figure 3.6: Acceleration plot from 2 IMUs

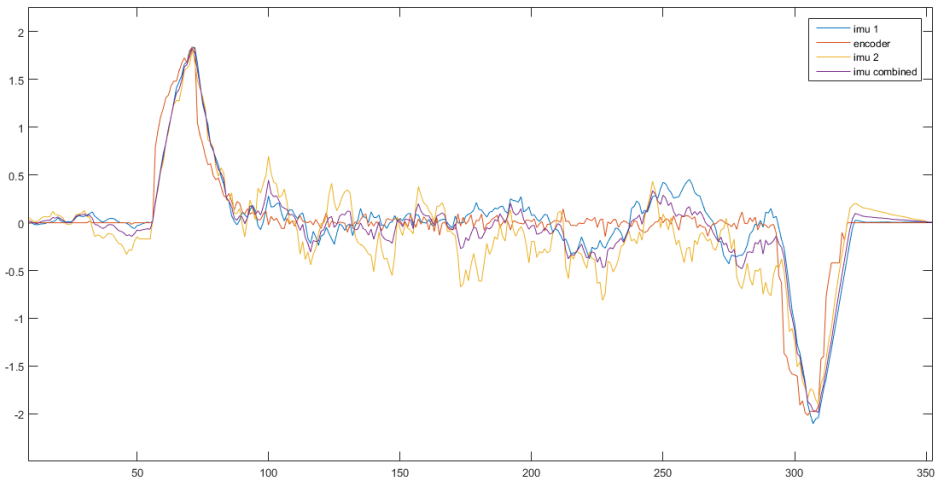

Figure 3.7: Dual IMU data combined with Kalman Filter

3.2.3 Acutuator feedback results

The no movement or no acceleration problem persists even with 2 IMUs and neural networks, to tackle this actuator feedback was used. Figure 3.8 shows changes in pwm that was used to filter out events when the robot will accelerate, under the assumption that enviromental forces are not enought to cause significant and persistant changes to the robot.

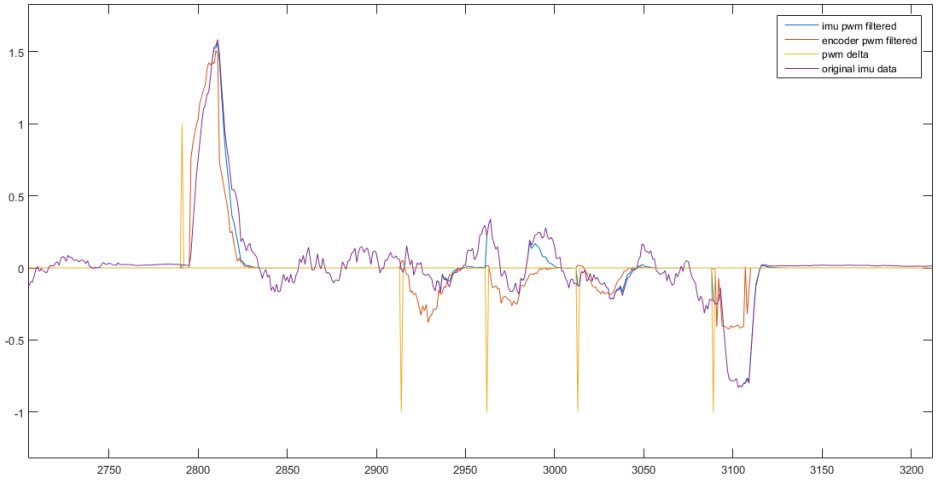

Figure 3.8: PWM Delta plot with direction of pwm change

The window function was then convolved with the pwm delta points to produce windows around the possible regions of acceleration and zero out other noises in the acceleration data outside the window intervals. Use of a half-bell window instead of a fixed length window yielded better results.

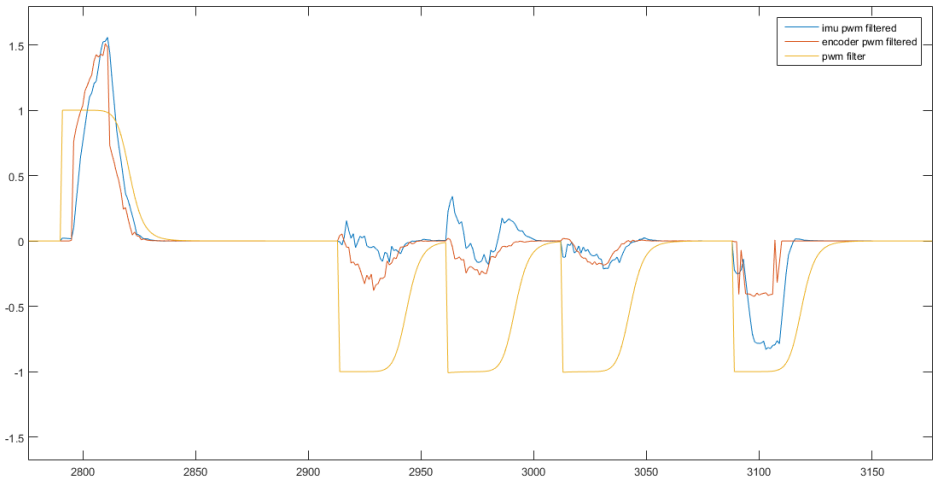

Figure 3.9: PWM Window function applied to filter noisy accelerations

3.2.4 Deep ANN filtering results

Combined with all the other modificafions as well as neural network training, the accuracy was improved much better than the original unfiltered data. As can be seen in figure 3.10, it is clear that over shorter periods of time, the INS drifts much less. Though drift cannot be eliminated completely, it is still very useful when combined with a auxilllary bounding navigaiton system such as VNS.

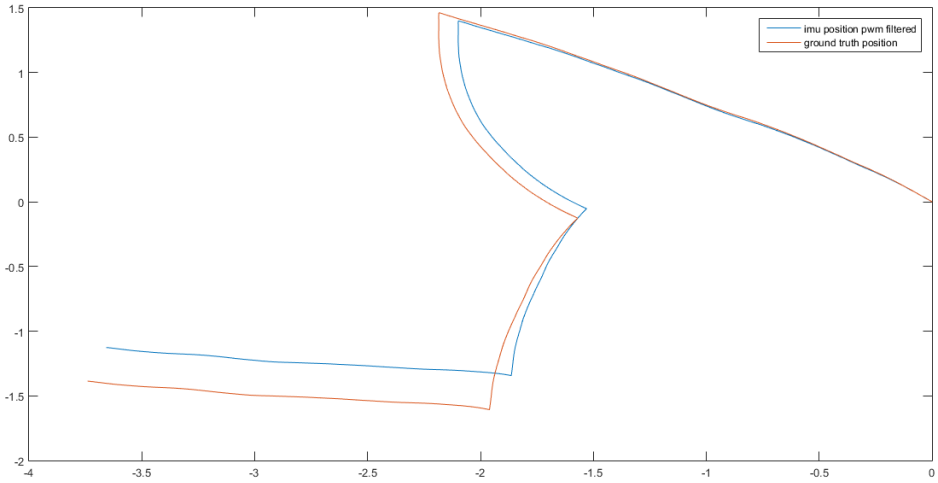

Figure 3.10: Result of all the INS modifications applied along with ANN training

3.2.5 INS benchmark

The collection on plots in figure 3.11 shows the final implementation of Modified INS run on a different untrained datasets of different lengths. It can be clearly seen that there is a trend towards increasing drift. But it is good to note that the drift for all datasets except one is limited to a small distance 10-20 cm within 500 timesteps which is equal to 10 seconds. This is a tolerable amount of error especially if a bounding navigation system can update the position within this period of time.

3.2.6 INS Conclusion

To conclude, it is noted that the INS system developed has reasonably low amount of error within short periods of time, this means that it can be used with Visual Navigation system, which is more accurate and doesn't drift, but is limited by availability of landmarks to track and data rate. Hence their combination will create a robust indoor navigation solution much like a GPS/INS solution. This same methodology can be used for AUVs as well, once extended to 3D space.

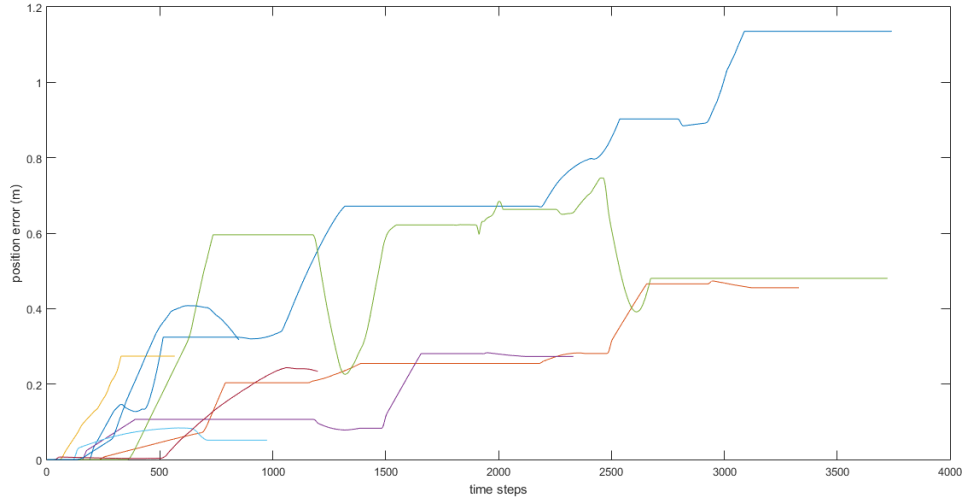

Figure 3.11: Benchmark of error plots over multiple test datasets

3.3 Vision based tracking with CNN

Many CNN architechtures have been proposed to track objects. Most of them work by running a classifier over different regions of the object and output the ones with higher probability score. These are relatively slow architechtures. The YOLO architechture works in a single pass, hence is much faster. This architecture was modified and used in this project.

3.3.1 Data Augmentation

Open Image databases used for training on common objects. The two databases used were Microsoft Common Objects in Context (COCO), and the MIT CSAIL Database of common objects. Training is done on 30 labels, a subset of the entire databases, based on the premise that map objects will be known before-hand. This allows use of a smaller network and reduce computation time.

The network is made robust by augmenting the data with the following methods.

- Rotation – Training done for different rotated images.
- Blur – Images are artificially blurred.
- Poor contrast – Lighting is artificially reduced.
- Occlusion – Images are partially cropped during and overlapped during training to produce occlusions.

Different methods used without much trade off in accuracy:

- Dropout - Reducing number of neurons by making them more independent. On each iteration, some of the neurons are randomly turned off.
- Singular Value Decomposition on Fully Connected Layer.

3.3.2 YOLO output format

The network predicts object bounding boxes above a threshold of object probability The bounding box coordinates are given as (x,y) of top left corner, size as (w,h) width, height, and object class predicted (c). This is shown in figure 3.12 which shows the bounding boxes predicted by the network.

Figure 3.12: YOLO CNN output

3.3.3 Transfer learning

Transfer learning is a method of using a pretrained network with a smaller dataset, with the low level features remaining the same, and only training the high level features. This reduces the training time, as well as make the network more accurate on the new dataset.

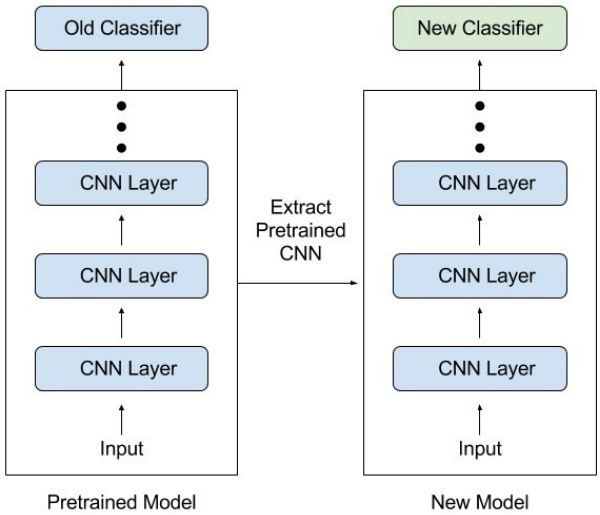

Figure 3.13: Transfer learning method for new datasets

Steps of Transfer learning, as show in figure 3.13

- Large amount of data is used to train for a long period.
- After training, the top layers are removed.
- New top layer is added, with random weights
- The lower layer weights are frozen(not trained anymore)
- Top layer weights are trained again with less data.
- This reduces the training time significantly, and network can adapt to new set of objects.

In visual navigation, the objects in a map are assumed to be known, and only 30 are used, so the modified YOLO network is retrained with the new set of target objects

3.3.4 Modifications to original YOLO architechture

A number of modifications were made to the original Fast YOLO architechture. The new architechture is reduced in size as only 30 object classes are targetted. The rest of the changes are discussed in section 3.4, they are to optimize it for hardware acceleration.

3.4 Hardware Acceleration of CNN

This section discusses two main ideas taken for optimized hardware acceleration of Convolutional Neural Networks on FPGAs. The first is network hyperparameter optimizations to use hardware friendly operations while not having any significant drop in accuracy; and the second is pipelining the computations across layers, to maximize the usage of parallel hardware blocks. [34] [35]

3.4.1 Hyperparameter Optimizations

A number of hyper-parameter optimizations were tried, most of were successful in converting the network to hardware friendly functions.

- **Quantization:** The network was initially trained with 32-bit single precision floating point datatypes. The weights and compute units were changed to use only 8-bit fixed point datatypes.[36], and [37] discuss more on this topic. The weights were quantized according to the method mentioned below, and the input data was also quantized during inference. This reduced the accuracy by 1.7 percent on the test database, but significantly reduced the computation time as the FPGA supports single clock Multiply and Accumulate (MAC) operations on 8-bit fixed point data, whereas floating point multiplications requires 6 clock cycles. Floating point multipliers also requires more sophisticated hardware which increases the resource usage by nearly 6 times. Hence this change was implemented on the CNN.

Type	Filters	Size/Stride	Output	Activation
Convolutional	32	3x3	224x224	ReLU
Maxpool	-	2x2/2	112x112	
Convolutional	64	3x3	112x112	ReLU
Maxpool	-	2x2/2	56x56	
Convolutional	128	3x3	56x56	ReLU
Maxpool	-	2x2/2	28x28	
Convolutional	256	3x3	28x28	ReLU
Maxpool	-	2x2/2	14x14	
Convolutional	512	3x3	14x14	ReLU
Convolutional	256	1x1	14x14	Tanh
Convolutional	512	3x3	14x14	Tanh
Maxpool	-	2x2/2	7x7	
Convolutional	1024	3x3	7x7	Tanh
Fully Connected	1000	All	1000	Tanh/ Softmax

Figure 3.14: Original YOLO Architechture

Method of Quantization: The weights of the convolutional layers were all software restricted to be within ± 2. After training, the data was then converted to be follow a (1,1,6) bit assignment pattern. Where the MSB is assigned as the sign bit, the 7th bit is assigned as the integral part, and the rest of the 6 bits are assigned as fractional parts. All the data was represented in this format, in case of overflows the value was clipped off at the maximum and minimum values, equal to ± 1.984375, the maximum quantization of this data representation is 0.015625.

- **Activation Function – Rectified Linear Unit (ReLU):** The ReLU activation function was originally adopted by researchers to work around the vanishing gradient problem found in lower layers of deep neural networks. Usually a continuously differentiable function like hyperbolic tangent (tanh) or sigmoid is preferred for stability. Replacing the activation function with ReLU for all convolutional layers increased the training time it took to reach 90 percent accuracy, but there onward it was faster than tanh training. It also achieved nearly identical accuracy score as compared to using tanh.

Type	Filters	Size/Stride	Output	Activation
Convolutional	32	3x3	128x128	ReLU
Avgpool	-	2x2/2	64x64	
Convolutional	64	3x3	64x64	ReLU
Avgpool	-	2x2/2	32x32	
Convolutional	128	3x3	56x56	ReLU
Avgpool	-	2x2/2	16x16	
Convolutional	256	3x3	16x16	ReLU
Avgpool	-	2x2/2	8x8	
Fully Connected	30	All	30	Tanh/ Softmax

Figure 3.15: Modified YOLO Architechture for Acceleration

Advantage of ReLU Hardware: ReLU is particularly inexpensive to implement in hardware. For datatypes that follow sign bit representation, it is simply a multiplexer with the select line tied to the sign bit, and the inputs being 0 and the number itself. In contrast, a tanh or sigmoid takes up much more resources, even optimized lookup table or piecewise linear models require much more resources than ReLU. Hence ReLU activation was used for all convolutional layers.

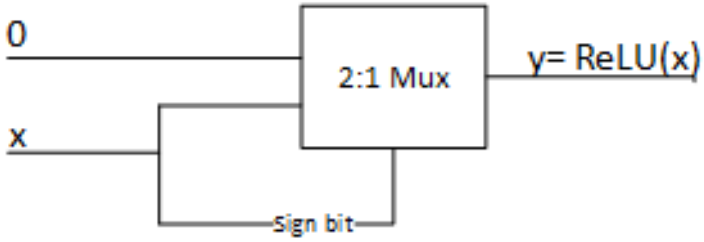

Figure 3.16: ReLU implementation on hardware

- **Average pooling:** Max pooling has been found to empirically perform better in many CNN implementations, [38] especially for larger pooling mask sizes. Testing average pooling on 2x2 and 4x4 mask with strides of 2, 3 and 4 yielded similar accuracies for both max pooling and average pooling, with maximum deviation of 1.2p percent across all the tested topologies and 0.4 percent for 2x2 mask which is most widely used.

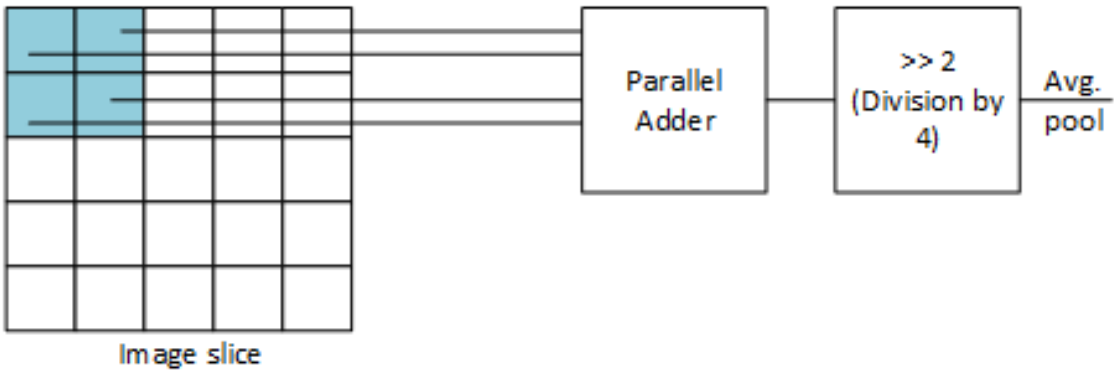

Figure 3.17: Average pooling implementation on harware

Average pooling in Hardware: Max pooling requires double variable input comparators, which are hardware intensive. Along with that, the comparisons have to happen sequentially or in a balanced tree, but it cannot be parallelized to work in a single clock cycle. For average pooling, a single large parallel adder unit followed by shifting approximation division operation can be used. When the mask size is 2x2 or 4x4, division is simply one right shift by 2 or 4 respectively, which requires much less hardware.

- **Network sizes in powers of 2:** In the FPGA, block memory units, registers, and logic modules are synthesized in blocks sizes in powers of 2. If network elements are initialized with a different size, say n, the closest power of 2 higher than n will be reserved, the extra reserved resources are hence not utilized, to ensure maximum utilization, the number of filters, pool mask sizes, input image size and datatypes were set to powers of 2. This also reduces the counter based state logic complexity as a power of 2 counter takes less resources than a mod n counter.
  Impact of size optimization: Reducing number of filters by more than 20 percent of the original network caused accuracy issues in the lower layers, with higher layers they could be reduced as much as 35 percent without any loss in accuracy. Where accuracy was dropping below tolerable margins, the number of filters were increased to the closest power of 2 which in turn increased accuracy. The effect of using 8 bit quantized data type has been discussed above.

3.4.2 Methods that didn't work

Here we present some of the methods that either dropped the accuracy significantly, or was not enough a trade off to use due to increase resources.

- **ReLU on fully connected layers:** Applying the ReLU activation on fully connected layers reduced the accuracy by over 7 percent, compared to tanh activation. Hence the tanh activations were computed on a host processor rather than on FPGA. It is

suspected that this is due to unstable behavior for higher dimensional data where the fully connected layers are used, further investigation is required for a better understanding.

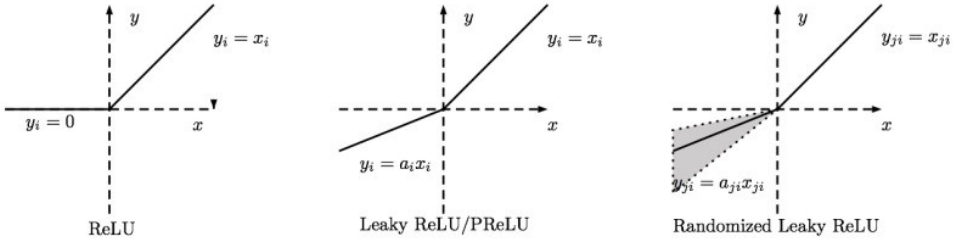

Figure 3.18: Leaky ReLU activation compared to ReLU

- **Leaky ReLU for better accuracy than ReLU:** The Leaky ReLU activation function considers using a reduced value of the negative input rather than clipping it to zero. The idea is to preserve some of the information of the negative region and hence require less number of filters. No significant improvement was found with usage of alpha value of 0.5 and 0.25, even with the same network size. Since Leaky ReLU takes more resources than ReLU but did not give significant improvement, the method was not used.

3.4.3 Pipelined Architecture

The architecture shown in figure 3.19 was developed to optimize the CNN computations, so all the hardware is used parallel, while also minimizing the memory usage to buffer the inputs and outputs.

Depending on the size of the network, the number of physical nodes in each layer is implemented, so as to balance the computational load between layers. This also considers the input data size reduction due to pooling. An example is given below to illustrate this.

Example network to optimize for pipelining: Suppose that we wish to balance the computational load between two consecutive layers (N and N+1) of the CNN, one takes the input image of size say SxS pixels, and layer N has M nodes; the next layer (N+1) will receive an input set of size (S/2) x (S/2) (Assuming pool mask size of 2), and say layer N+1 has P nodes.
Total number of convolutions in Layer N = S x S x 1 x M
Total number of convolutions in Layer N+1 = (S/2) x (S/2) x M x P

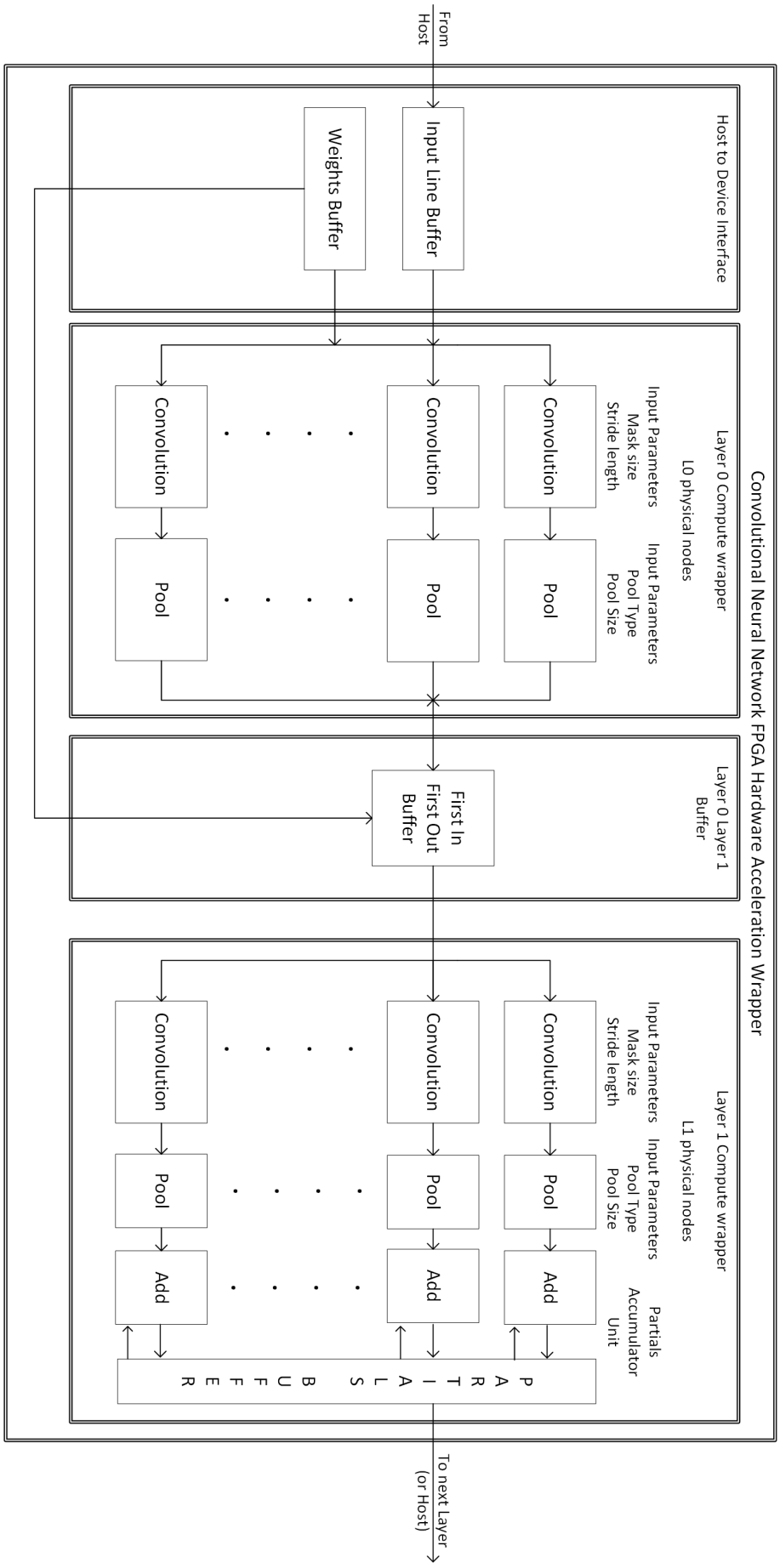

Figure 3.19: Pipelined Architechture for FPGA Implementation of CNN

The ratio of physical nodes in layer N and layer N+1 should be equal to ratio of the total convolutions in each layer. Therefore

$$\frac{L_N}{L_{N+1}} = \frac{S \times S \times 1 \times M}{\frac{S}{2} * \frac{S}{2} * M * P} = \frac{4}{P}$$

This can be generalized as

$$\frac{L_N}{L_{N+1}} = \frac{D^2 * R}{P}$$

D can also be computed as D = Q + S - 1, where Q is the pool mask size and S is the stride size.

The network sizes should hence be designed with pipelining requirements in mind. The figure 3.20 shows a simple example case of a balanced computation pipeline.

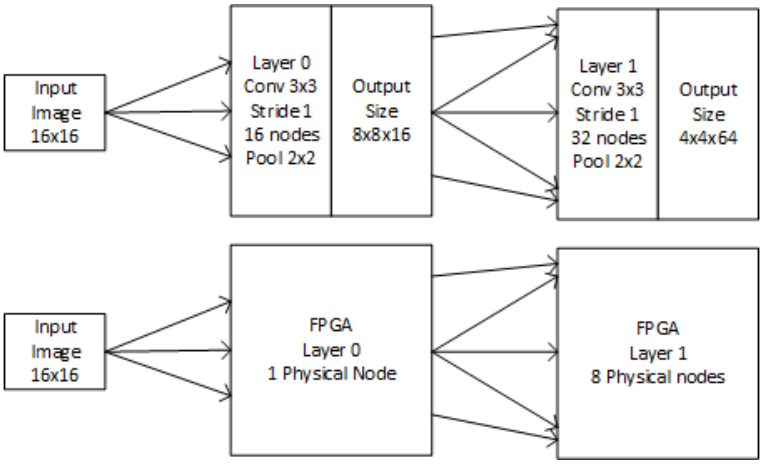

Figure 3.20: Example of a balnace computation pipeline

3.5 VNS simulation results

This section presents results of Visual Navigation System's simulation results obtained from a simulation platform. Due to lack of time, the results are only a first draft and not comprehensive over many iterations with different situations and simulated environmentals conditions. Nevertheless, it offers some important insight to the performance of the system.

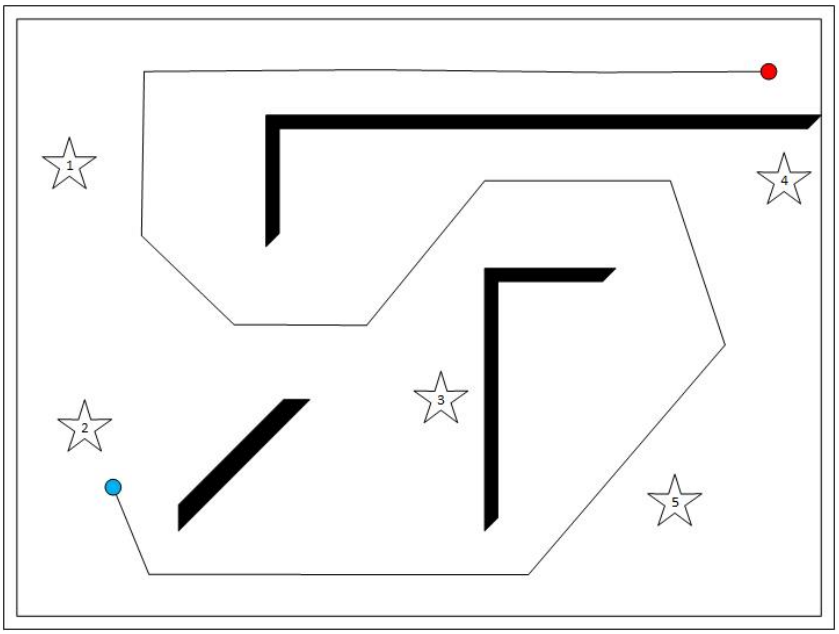

Figure 3.21: VNS Simulation ground truth overlaid on the map

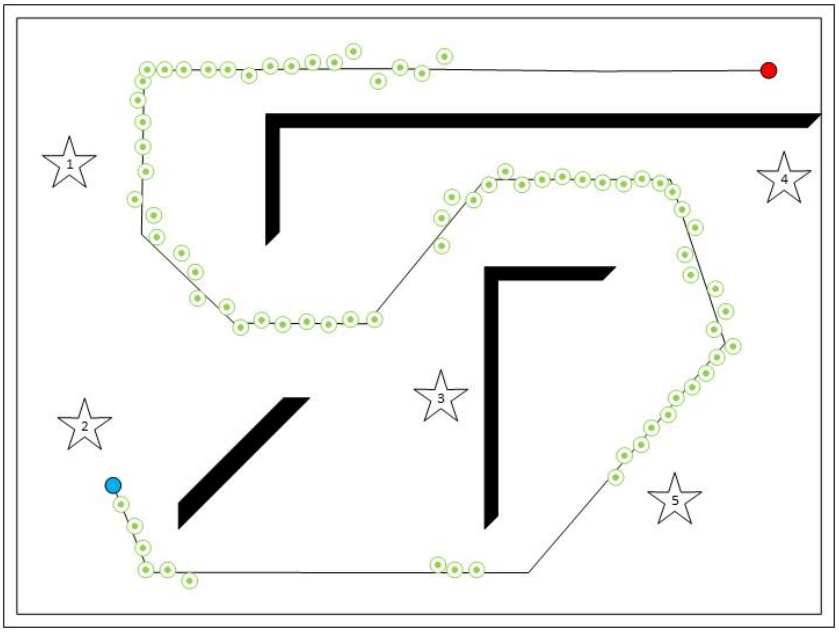

Figure 3.22: VNS Simulation results overlaid on the map

As can be seen from figures 3.21 and 3.22, it is clear that as the agent moves through the map, it's accuracy increases as it gets closer to the landmarks.It is noted that the VNS cannot work without line of sight, as it loses view of the landmarks, it isn't able to localize any more. This is where a inertial navigation system would aid it to provide navigation information till it is able to find new landmarks. Both the data can then be combined to make an accurate GPS denied navigation system.

Chapter 4

Conclusion

The objectives completed in this thesis mainly comprised of Inertial and Visual Navigation systems as separate subsystems. The INS system was tested on a hardware platform with multiple datasets very in a comprehensive manner. The VNS system was only tested on a limited simulation environment.

Inertial Navigation system with 2 IMUs was integrated and tested on real world system, with ground truth obtained from wheel encoders. A benchmark was made over multiple runs to get the maximum drift over a given time period. The accuracy was proven with this hardware setup and can be replicated with the shared software.

Visual Navigation system was implemented in simulation, tracking only known objects with a Convolutional Neural Network and perspective projection was used to get the robot's coordinates. Visual Navigation CNN software was accelerated in FPGA hardware with a pipelined architecture, as well as best known methods were established for some of the hardware friendly operations, and also some methods were found to not work as well as expected.

Lastly, a software framework was made to extend the system with Robotic Operating System, since the original hardware for INS was integrated with ROS, new subsystems like Visual Navigation System can be integrated to the existing work more easily.

The code for most of the submodules in this project can be found at:
https://github.com/Dipamc77/InertialNav

Scope for Further Research

The project has a very wide range of applications, and hence has a lot of scope for improvements. The following ideas are left as future work:

- VNS to be implemented on real world harware.
- VNS to be integrated with INS in simulation and real world system.
- System to be augmented with more sensors.

• VNS to be extended to unknown objects for Visual SLAM.
• System to be extended to 3D space for autonomous underwater/aerial vehicles.

The figure 4.1 depicts a conceptual diagram to merge the VNS and INS algorithms which will yield much better accuracy. Similar discussion is done in [39]

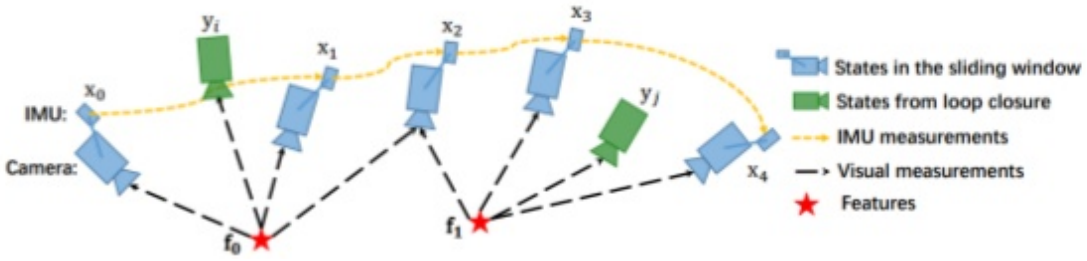

Figure 4.1: Visualization of the VNS and INS fusion methodology

The idea is to use VNS as a bounding system, since it has a slower update rate and dependent of visual features, while a faster but less accurate INS, which drifts over long periods, but is accurate in short intervals would compensate for the VNS during the outage period.

To conclude, the project has very broad scope for further research, and this thesis should serve as a stepping stone towards further developments in this field.

References

[1] G. N. Desouza and A. C. Kak, "Vision for mobile robot navigation: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 237–267, Feb 2002.

[2] P. D. Groves, *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech house, 2013.

[3] R. G. Brown, P. Y. Hwang *et al.*, *Introduction to random signals and applied Kalman filtering*. Wiley New York, 1992, vol. 3.

[4] D. H. Won, S. Chun, S. Sung, Y. J. Lee, J. Cho, J. Joo, and J. Park, "Ins/vslam system using distributed particle filter," *International Journal of Control, Automation and Systems*, vol. 8, no. 6, pp. 1232–1240, 2010.

[5] S. Godha, G. Lachapelle, and M. E. Cannon, "Integrated gps/ins system for pedestrian navigation in a signal degraded environment," in *ION GNSS*, vol. 2006, 2006.

[6] D. Bhatt, P. Aggarwal, V. Devabhaktuni, and P. Bhattacharya, "A novel hybrid fusion algorithm to bridge the period of gps outages using low-cost ins," *Expert Systems with Applications*, vol. 41, no. 5, pp. 2166–2173, 2014.

[7] X. Ning, M. Gui, Y. Xu, X. Bai, and J. Fang, "Ins/vns/cns integrated navigation method for planetary rovers," *Aerospace Science and Technology*, vol. 48, pp. 102–114, 2016.

[8] Q. Pan, F. Yang, L. Ye, Y. Liang, and Y.-m. Cheng, "Survey of a kind of nonlinear filters-ukf," *Control and Decision*, vol. 20, no. 5, p. 481, 2005.

[9] S. Omari, M. Bloesch, P. Gohl, and R. Siegwart, "Dense visual-inertial navigation system for mobile robots," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2634–2640.

[10] Q.-h. Meng, Y.-c. Sun, and Z.-l. Cao, "Adaptive extended kalman filter (aekf)-based mobile robot localization using sonar," *Robotica*, vol. 18, no. 5, pp. 459–473, 2000.

[11] K. D. Sebesta and N. Boizot, "A real-time adaptive high-gain ekf, applied to a quadcopter inertial navigation system," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 1, pp. 495–503, 2014.

[12] A. Tal, I. Klein, and R. Katz, "Inertial navigation system/doppler velocity log (ins/dvl) fusion with partial dvl measurements," *Sensors*, vol. 17, no. 2, p. 415, 2017.

[13] R. Madhan, E. Desa, S. Prabhudesai, L. Sebastião, A. Pascoal, E. Desa, A. Mascarenhas, P. Maurya, G. Navelkar, S. Afzulpurkar *et al.*, "Mechanical design and development aspects of a small auv-maya," in *7th IFAC Conference MCMC2006*, 2006.

[14] M. Dunbabin, J. Roberts, K. Usher, G. Winstanley, and P. Corke, "A hybrid auv design for shallow water reef navigation," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 2105–2110.

[15] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, "The vslam algorithm for robust localization and mapping," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 24–29.

[16] S. Godha, "Performance evaluation of low cost mems-based imu integrated with gps for land vehicle navigation application," *UCGE report*, vol. 20239, 2006.

[17] W. Li and J. Wang, "Effective adaptive kalman filter for mems-imu/magnetometers integrated attitude and heading reference systems," *The Journal of Navigation*, vol. 66, no. 1, pp. 99–113, 2013.

[18] J.-K. Shiau, C.-X. Huang, and M.-Y. Chang, "Noise characteristics of mems gyro," □□□□□□, vol. 15, no. 3, pp. 239–246, 2012.

[19] O. S. Salychev, *Applied Inertial Navigation: problems and solutions*. BMSTU press Moscow, Russia:, 2004.

[20] J. Yi, J. Zhang, D. Song, and S. Jayasuriya, "Imu-based localization and slip estimation for skid-steered mobile robots," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 2845–2850.

[21] I. Colomina, M. Giménez, J. Rosales, M. Wis, A. Gomez, and P. Miguelsanz, "Redundant imus for precise trajectory determination," in *Proceedings of the 20th ISPRS Congress, Istanbul, Turkey*, vol. 1223. Citeseer, 2004, p. 17.

[22] M. T. Hagan, H. B. Demuth, M. H. Beale *et al.*, *Neural network design*. Pws Pub. Boston, 1996, vol. 20.

[23] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *CoRR*, vol. abs/1511.08458, 2015.

[24] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, and G. Wang, "Recent advances in convolutional neural networks," *CoRR*, vol. abs/1512.07108, 2015.

[25] R. Girshick, "Fast r-cnn," *arXiv preprint arXiv:1504.08083*, 2015.

[26] J. Shi *et al.*, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

[27] C. F. Olson, "Maximum-likelihood template matching," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2. IEEE, 2000, pp. 52–57.

[28] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[29] K.-J. Yoon and I.-S. Kweon, "Artificial landmark tracking based on the color histogram," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 4. IEEE, 2001, pp. 1918–1923.

[30] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[31] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," in *Advances in neural information processing systems*, 2006, pp. 1161–1168.

[32] H. Qi and J. B. Moore, "Direct kalman filtering approach for gps/ins integration," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 2, pp. 687–693, 2002.

[33] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[34] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2015, pp. 161–170.

[35] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010, pp. 257–260.

[36] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[37] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[38] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2559–2566.

[39] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *The International Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, 2011.