B.Eng. Dissertation

Bumblebee AUV/ASV Development Work

By

Alex Philipose John

Department of Computer Science

School of Computing

National University of Singapore

2016/17

B.Eng. Dissertation

Bumblebee AUV/ASV Development Work

By

Alex Philipose John

Department of Computer Science

School of Computing

National University of Singapore

2016/17

Project No: H096220
Advisor: Prof. David Hsu
A/P Marcelo Ang
Deliverables:
1 Report

Abstract

In this report I present the design and development of core software components that run on board an Autonomous Underwater Vehicle (AUV) and Autonomous Surface Vessel (ASV). To achieve the ultimate goal of demonstrating autonomous launch and recovery (LARS) of the AUV from the ASV, we first develop the navigation and control system. First, a suitable underwater simulation stack is developed and the derivation of the hydrodynamic model is detailed. Second, the simulation stack is used to test the control system and a simple proof of concept neural network based real time PID tuner is developed. This enables the PID controller to adapt its constants to changing environmental conditions, particularly wind. Off the shelf sensors used in inertial navigation is benchmarked and compared. The inertial navigation system used on two separate vehicles was dismantled and re-developed as an Error State Kalman filter that dynamically adapts to either GPS, Camera, USBL or DVL observations. Real world test results are showcased and analyzed.

Subject Descriptors: Keywords:

Kalman filters and hidden Markov models
Physical simulation
Robotics

robotics, kalman, inertial navigation, control systems, underwater simulation

Implementation Software and Hardware:
Linux x64, Armv8hf, C++, BumblebeeAUV 3.5, Bumblebee ASV 1.0

Acknowledgements

I would like to express my deepest appreciation and thanks to my supervising professors, Marcelo Ang and David Hsu for their guidance and time.

Working with autonomous marine systems has been an amazing experience and for which I would like to thank my team, Bumblebee Autonomous Systems. I would like to thank in particular, Grace Chia and Goh Eng Wei for their excellent leadership and guidance for the past four years.

In addition, a very big thank you the School of Computing and Faculty of Engineering for their sponsorship and support.

Table of Contents

Title **1**

Abstract **2**

Acknowledgements **3**

1 Introduction **1**

2 Frame Definition and Transforms **4**
2.1 Quaternions and Notation 4
2.2 Frames 6
2.2.1 world 6
2.2.2 local 6

3 Underwater Simulation **7**
3.1 Simulation of an IMU 7
3.1.1 Angular Velocity 7
3.1.2 Acceleration 9
3.1.3 Magnetometer 9
3.2 Simulation of a DVL 9
3.3 AUV Dynamics 11
3.3.1 Constants 12
3.3.2 Allocation Matrix 12
3.3.3 Thruster Measurement 13
3.3.4 Mass and Inertia Matrix 16
3.3.5 Coriolis and Centripetal Matrices 17
3.3.6 Hydrodynamic Damping Matrix 17
3.3.7 Buoyancy and Force due to Gravity 18
3.3.8 Currents and Wind 19

4 Control System **20**
4.1 PID Controller 20
4.1.1 Proportional Constant 21
4.1.2 Integral Constant 21
4.1.3 Derivative Constant 22
4.1.4 Tuning and Real World Use 22
4.1.5 Tuning UI 23
4.2 Adaptive PID Controller 24
4.2.1 Feature Extraction 24
4.2.2 Neural Network Tuner 26

5 Navigation **29**
5.1 Filter Design and Architecture 29
5.2 Comparison between IMUs 30
5.2.1 STIM300 30
5.2.2 Sparton AHRS-8 31
5.2.3 SBG Ellipse 32
5.3 Comparison between GPS Antennae 32
5.4 Preprocessing GPS Data 34
5.5 Kalman Filtering 35
5.5.1 Kinematic Equations 36
5.5.2 Error State Kalman Filter 37
5.6 GPS Fault Detection 42
5.7 Results 43

6 Conclusion **50**

Chapter 1

Introduction

Exploring the underwater world is a hard challenge. The sea is harsh and unpredictable. It is a very risky process to send humans underwater, due to the extreme pressures and temperatures involved. Although not safe for humans, it is completely safe for robots to do repetitive and time consuming work out in the sea. Usual tasks include survey operations, sample collection, blackbox recovery for downed aircraft and biological and research trips.

Recently there has been a lot of interests in mapping out the ocean floor, mining for underwater resources and other long term commitments. Autonomous vessels are sought after in this industry as they simultaneous make it safer for the man power involved and speed up the process. One of the biggest drawbacks of AUVs has always been its battery capacity and time underwater. Certain missions spent half of the battery pack just getting in position from either a ship or shore.

This is where the motivation for using autonomous surface craft comes in

as well. Ships are expensive to maintain and to run, in addition to further risk of personnel. Surface craft can carry an AUV to its position, maintain position via use of GPS and provide transducers for navigation baselines, data transfer to shore and even charging.

Bumblebee ASV was purchased almost years ago to take part in the Maritime RobotX 2016 in Hawaii. The boat will be taking part in RobotX 2018 in which launch and recovery of a submersible is a major factor. Not considering competition, the plan was to work on it and achieve complete autonomous launch and recovery by the end of this year. Due to competition schedules, manufacturing and design and other constraints, the timeline was pushed back to December 2018.

There are other teams that are also trying to achieve launch and recovery, particularly from companies like Kongsberg and other university groups[**rauch2008ship**].

Navigational accuracy is a determining factor in if we are confident of letting the AUV go without a tether. Of course, a tether has other uses such as streaming high quality camera data. Companies such as Teledyne and Sondardyne (amongst others) have off the shelf solutions that provide high navigational accuracy. They however use a fiber optic gyro and other sensors[1] such as USBL (ultra short baseline) and LBL (long baseline). LBL provides millimetre precision[**whitcomb1999combined**] but it takes very long to set up and requires transponders to be set and calibrated at precise

[1]From the product datasheet

locations.

The option to interface USBL with our existing filter will be provided for the future but for us dead reckoning based positioning accuracy is important to remain competitive with respect to cost amongst other considerations such as size and weight.

One of the other biggest problems that we faced in Hawaii was the wind. Our boat only had two thrusters back then and we could not station keep. To counter that, we have moved to vectored thrust and that is also the motivation behind exploring adaptive schemes of control for the boat. University of Florida took the same course of action by going vectored thrust with 4 trawling motors[**graynavigator**] and using a Model Reference Adaptive Controller. UF has also attempted the use of an underwater vehicle from their boat[**gray2016anglerfish**].

As our boat gets quicker to competition more topics can be explored tested and implemented for competition as well as real world scenarios.

Chapter 2

Frame Definition and Transforms

2.1 Quaternions and Notation

Quaternions form a system that extends the complex number system. Complex numbers of unit length can encode rotations in the 2D plane. Likewise, extended complex numbers can encode rotations in 3D space. Quaternions are represented by a real part (scalar) and the imaginary part [**Eberly2002**].

$$Q = q_w + q_x i + q_y j + q_z k$$

$$i^2 + j^2 + k^2 = -1 = ijk$$

There are two frequently used representations for quaternions[1]: Hamil-

[1]There exist others: ESA, ISS, etc;

tonian and JPL. Hamiltonian quaternions are usually denoted as $\begin{bmatrix} q_w & \mathbf{q}_v \end{bmatrix}^T$ with the real part in the front. The corresponding JPL notation is $\begin{bmatrix} \mathbf{q}_v & q_w \end{bmatrix}^T$. The Hamiltonian notation is used for quaternions throughout this report as it is the represented used by the C++ linear algebra library Eigen and Robot Operating System both of which are used heavily onboard our platforms.

Compared to rotation matrices that can also encode rotation in 3D space which requires 9 doubles to store the corresponding quaternion representation only requires 4. This offers significant savings in bandwidth when used streaming high frequency odometry data.

Composition of frames (combining two rotations) usually requires matrix normalization to hold the rotation matrix orthogonality constraint $RR^T = I$. When composition is performed using numerical methods, the constraint is usually broken by rounding errors particularly so when using single precision floating point arithmetic. Matrix renormalization is a computationally expensive process requiring computation of the singular value decomposition and replacing the singular values with ones[**Horn2008**] . On the other hand, quaternion normalization is an inexpensive process with most libraries providing the functionality out of the box:

$$q_U = \frac{q}{||q||}$$

$$||q|| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2}$$

2.2 Frames

2.2.1 world

World fixed frame with its z-axis pointing upwards. The world frame is also aliased as the map frame. The world frame is globally referenced with the help of GPS and time variant drift is not expected or at least, expected to be negligible

2.2.2 local

The local body frame is the reference to which any strap down inertial navigational sensors are mounted to the vehicle with.

Chapter 3

Underwater Simulation

An existing underwater simulator[**prats2012open**] which has not been maintained was fixed up and used for simulation of the AUV and ASV. Underwater Simulator (UWSim) is written in C++ using the OpenSceneGraph libraries. It has existing integration with Robot Operating System and Gazebo.

3.1 Simulation of an IMU

3.1.1 Angular Velocity

UWSim does not come with a functioning simulated IMU sensor. We can however obtain the orientation of the model from OSG as a quaternion. This quaternion can then be perturbed with normally distributed noise with a specified standard deviation.

The quaternion orientation q is a curve $q : I \rightarrow \mathbb{S}^3 \in \mathbb{R}^4$ where the quaternion is represented as a 4 dimensional vector. We can hence compute

the time derivative as explained in [**chou1992quaternion**]

$$\dot{q}(t) = \frac{d}{dt}q(t) = \lim_{h\to 0} \frac{q(t+h) - q(t)}{h} \tag{3.1}$$

and for small δ can be approximated by

$$\frac{q(t+\delta) - q(t)}{\delta}$$

Since $\mathbb{S}$ is a Lie group the angular velocity associated with $q(t)$ as

$$\dot{q}(t) = \frac{1}{2}q(t) \star \begin{bmatrix} 0 \\ \omega(t) \end{bmatrix} \tag{3.2}$$

where $\star$ is quaternion multiplication. Rearranging we can see that

$$\omega(t) = \Im(2\bar{q(t)} \star \dot{q}(t))$$

where $\Im$ extracts the imaginary part out (vector) and $\bar{q}$ represents the conjugation.

Substituting (3.1) we have the equation

$$\begin{aligned} \omega(t) &= \Im(2\bar{q(t)} \star \lim_{h\to 0} \frac{q(t+h) - q(t)}{h}) \\ &= \Im(2 \lim_{h\to 0} \frac{\bar{q(t)} \star q(t+h) - \bar{q(t)}q(t)}{h}) \\ &= \Im(2 \lim_{h\to 0} \frac{\bar{q(t)} \star q(t+h)}{h}) \end{aligned}$$

where for small δ we have $\omega(t) = 2\bar{q(t)} \star \frac{q(t+\delta)}{2}$

For numerical computation the quaternion state without noise is used and normally distributed noise is added to the angular velocity obtained later.

3.1.2 Acceleration

The transform for a rendered object in the world frame can be obtained from the OpenSceneGraph API. The acceleration can then be computed by double numerical differentiation from two different positions of the object for small δt

Normally distributed guassian noise is introduced to the calculated vector.

3.1.3 Magnetometer

A magnetometer measures the direction and field strength of a magnetic field. Given a world frame position fix in LLA[1] and the time, the corresponding field vector is obtained using the World Magnetic Model. The vector is rotated to the local frame and perturbed with a scale and bias factor.

Magnetometers are often inaccurate and affected by many factors, most commonly ferromagnetic structures and even the electromagnetic fields generated by the carrying vehicle. This feature is not implemented and is left for future improvement.

3.2 Simulation of a DVL

A DVL is a sensor that measures the speed relative to the seabed or relative to the flow of water (Acoustic Doppler Current Profiler mode). The sensor reading that it gives is relative to the body frame of the vehicle. We can get

[1]Latitude, Longitude, Altitude (above sea level)

the transform for the position of the object relative to the OSG world.

$$v(t) = \mathbf{R}_w^b v(t)$$

The velocity can be obtained by numerical differentiation with added guassian noise.

The sensor measures velocity by sending out four beams of sound at an angle θ from the center of the sensor head. This is called the Janus angle and it varies from as little as 10 deg to 30 deg. The matrix that is formed from the system of equations that relate beam velocities to ENU velocities is given by[**Gilcoto2009**]

$$\mathbf{T}_{DVL} = \begin{bmatrix} -\sin\theta & 0 & -\cos\theta \\ \sin\theta & 0 & -\cos\theta \\ 0 & \sin\theta & -\cos\theta \\ 0 & -\sin\theta & -\cos\theta \end{bmatrix} \tag{3.3}$$

and given velocity v in body frame (ENU) we can obtain beam velocities by multiplication

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = T_{DVL} v$$

We can transform a 4 beam solution back to the ENU frame using a least squares fit $(T^T T)^{-1} T^T b$

3.3 AUV Dynamics

The calculation of AUV dynamics is important as its lets us test the control system within the simulator. It also lets the mechanical team approximate the dynamics of the AUV by using calculated constants from SolidWorks.

The dynamic model of the AUV as represented by [**yuh1995underwater**] is given by

$$M\dot{V} + C(V)V + D(V)V + G = \tau \tag{3.4}$$

where V is the velocity state vector $\begin{bmatrix} \mathbf{v} & \omega \end{bmatrix}^T$, M is the mass inertia matrix, $C(V)$ is the coriolis and centripetal matrix and $D(V)$ is the hydrodynamic damping matrix. τ is the external force and torque input vector.

3.3.1 Constants

The constants for BBAUV 3.5 as obtained from SolidWorks and measurements in the lab are given below

$$M = 56$$

$$n_{act} = 8$$

$$\mathbf{C_g} = \begin{bmatrix} 0.47 & 0.68 & 0.85 \end{bmatrix}^T$$

$$I = \begin{bmatrix} 4.86 & 0.01 & 0.01 \\ 0.01 & 4.59 & 0.12 \\ 0.01 & 0.12 & 0.94 \end{bmatrix}$$

$$\nabla = 0.02m^3$$

C_g is the center of gravity, I holds the moments of inertia, ∇ the volume displaced and we have 8 actuators. It is observed that the center of gravity is off but the moments of inertia matrix is symmetric around the diagonal with minimal off diagonal terms. If there are significant off diagonal terms any rotational motion will be unstable[**nahon1996simplified**].

3.3.2 Allocation Matrix

Allocation matrix is $n_{dof} \times n_{act}$ and calculates the force that the AUV exerts in its 6 dof: surge, sway, heave, roll, pitch and yaw as a response to actuator input.

l_1	0.250
l_3	0.500

Table 3.1: Thruster positioning in m

$$\mathbf{L} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -l_3 & l_4 & -l_3 & l_4 & 0 & 0 \\ 0 & 0 & -l_5 & -l_5 & l_6 & l_6 & 0 & 0 \\ -l_1 & l_2 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{3.5}$$

where $l_1 = l_2$ is the distance from the center line of the AUV to Videoray thrusters and $l_3 = l_4$ is the distance from the cetnter line of the AUV to the seabotix thrusters. The numbers obtained from SolidWorks is given in the table 3.1

The force obtained in each of the dof can then be computed if we have thruster input vector u as $\mathbf{L}u$.

3.3.3 Thruster Measurement

On the AUV and ASV we cannot command the actuator based on the amount of force it outputs but rather using a mapping on the discrete integer interval between [-1600, 1600] for surge thrusters and [-3200, 3200] for others. This corresponds to the duty cycle of the output PWM. A thruster measurement jig was constructed to measure the bollard pull or the single axis force ob-

tained from the thruster. The obtained results in were then curve fitted to obtain the input vs output (force) curve.

We can observe that

- The thrusters are highly non-linear
- Curve fitting for both forward and reverse as a single function gives bad fit with R value < 0.92

Thus a piecewise function was used to model the thrusters and is given below

$$F_v(x) = \begin{cases} -4.308 \times 10^{-10}x^3 - 1.214 \times 10^{-6}x^2 + 0.0003757x & x \leq 0 \\ -1.49 \times 10^{-10}x^3 - 3.88 \times 10^{-7} + 0.0035x & x > 0 \end{cases} \tag{3.6}$$

and for seabotix thrusters

$$F_s(x) = \begin{cases} 2.5 \times 10^{-11}x^3 - 1.84 \times 10^{-08}x^2 + 1 \times 10^{-5}x & x \geq 0 \\ -2.72 \times 10^{-12}x^3 - 7.32 \times 10^{-08}x^2 & x < 0 \end{cases} \tag{3.7}$$

The thrust curves are visualized in figures 3.1 and 3.2. It is immediately obvious that the Videoray Pro 4 is significantly more non-linear than the Seabotix thruster.

For use in the dynamic model of the AUV, we have thruster command vector u which is the output from the PID controller. The above obtained functions are then mapped column wise onto the vector before multiplication with the allocation matrix.

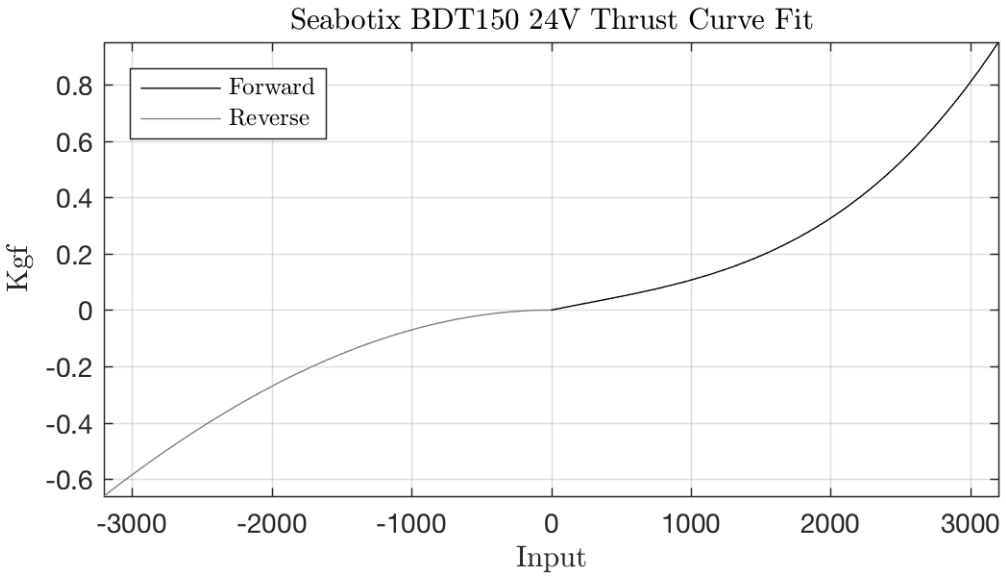

Figure 3.1: Seabotix Thrust Curve

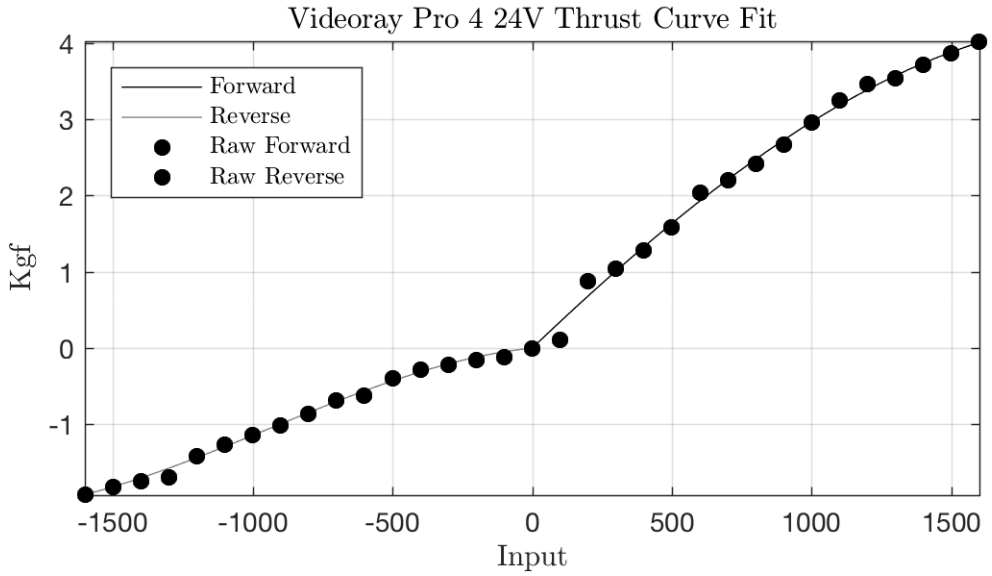

Figure 3.2: Videoray Thrust Curve

3.3.4 Mass and Inertia Matrix

The mass and inertia matrix M is composed of M_{rb} the mass and inertia for a rigid body and M_A, a hydrodynamic added mass.

$$M = M_{rb} + M_A$$

$$\mathbf{M}_{rb} = \begin{bmatrix} m & & & & & \\ & m & & & [-m * C_g]_\times & \\ & & m & & & \\ & & & I_{xx} & I_{xy} & I_{xz} \\ & [m * C_g]_\times & & I_{yx} & I_{yy} & I_{yz} \\ & & & I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

where C_g is the center of mass, and $_\times$ is the anti skew symmetric matrix.

We can approximate the hydrodynamic added mass as half of the original mass. According to literature[**Vervoort2009**] the value lies between 10% to 100% of the corresponding parameters in M_{rb} ; thus

$$M_A = \begin{bmatrix} \frac{m}{2} & & & & \\ & \frac{m}{2} & & & \\ & & \frac{m}{2} & & \\ & & & 0 & \\ & & & & \ddots \end{bmatrix}$$

3.3.5 Coriolis and Centripetal Matrices

We can then compute the coriolis matrix which is used to analyze forces applied on the rigid body under rotation.

$$C(V) = \begin{bmatrix} 0 & \dots & 0 & \\ \vdots & & & -[s_1]_\times \\ & & & \\ & & & \\ & -[s_1]_\times & & -[s_2]_\times \\ & & & \end{bmatrix}$$

where

$$s_1 = Mv + [mC_g]_\times \omega$$

$$s_2 = [-mC_g]_\times v + I\omega$$

I is the second order inertia matrix and not the identity matrix. The derivation of the Coriolis matrix is given in [**Vervoort2009**].

3.3.6 Hydrodynamic Damping Matrix

The hydrodynamic damping matrix which contains the drag and lift forces can be separated into a linear and quadratic term but we can discard the lift force assuming our vehicle or the current its facing is not very high.

Both the linear and quadratic terms can be represented as diagonal 6x6 matrices where

$$X = \frac{1}{2}\rho C_d A_f$$

$$D(V) = \begin{bmatrix} X & & & & 0 \\ & Y & & & \\ & & \ddots & & \\ & & & K & \\ & & & & \ddots \end{bmatrix} + \begin{bmatrix} X_{u|u}|v_x| & & & & \\ & & \ddots & & \\ & & & & \\ & & & K_{p|p}|\omega_x| & \\ 0 & & & & \ddots \end{bmatrix}$$

3.3.7 Buoyancy and Force due to Gravity

$$W = mg$$

$$B = \rho g \nabla$$

where d is the density, r is the radius and ∇ is the volume displaced by the AUV. The force associated G can be calculated by[**Vervoort2009**]

$$G = \begin{bmatrix} R_b^w \begin{bmatrix} 0 & 0 & W \end{bmatrix}^T - R_b^w \begin{bmatrix} 0 & 0 & B \end{bmatrix}^T \\ c_g \times \begin{bmatrix} 0 & 0 & W \end{bmatrix}^T - c_g \times \begin{bmatrix} 0 & 0 & B \end{bmatrix}^T \end{bmatrix}$$

Bumblebee AUV is meant to be 1% positively buoyant. However, from examination of the buoyant force and gravity acting on the rigid body numbers from SolidWorsk it is negatively buoyant by 348.27N[2]. The mismatch is often corrected with buoyancy foam. We can hence tune the volume displaced in the simulator or work with the depth controller to maintain depth. In the default state however, the vehicle is expected to sink.

[2]Taking volume displaced as 0.02 m^3

3.3.8 Currents and Wind

We can extend (3.4) and direct the current velocities obtained from the simulator as[**do2009control**]. The disturbances can be mapped directly to body frame for ease of computation when testing control systems. However, when teh disturbances are obtained from UWSim, we need to transform it from world to body frame before applying it to eq (3.4)

$$M\dot{V} + C(V)V + D(V)V + G = \tau_{act} + \tau_{wind} + \tau_{wave} \tag{3.8}$$

Chapter 4

Control System

4.1 PID Controller

Both Bumblebee AUV and ASV run on closed loop Proportional Integral Derivative controllers. There is one PID loop per degree of freedom. The thrusters are shared according to the thrust allocation matrix shown in (3.5).

The PID equation can be described mathematically as [**astrom2010feedback**]

$$O(t) = K_p e(t) + K_d \frac{de}{dt} + K_i \int_0^t e(\tau) d\tau \tag{4.1}$$

where τ is the variable of integration (integrates over the history) and $e(t)$ is the error associated $e = s - i$ where s is the setpoint and i is the sensor input.

4.1.1 Proportional Constant

The proportional constant K_p produces basic output in response to error. It controls the proportional gain. If the gain is too high we can see heavy oscillation. If the gain is too low, the controller fails to achieve its setpoint.

4.1.2 Integral Constant

The integral term is used to control the magnitude and duration of the error, often steady state error. If a controller fails to achieve the setpoint due to physical or actuator limits or stabilizes with an offset to the setpoint, the integral controller accumulates the error every time step and increases the output value. The integral term is clamped to a realistic value to account for physical limits on the actuators.

Windup protection

Integral windup occurs when a large change in setpoint occurs. Real world actuators do not act instantaneously and accumulate integral term errors even before they reached the setpoint. The previously implemented clamp counter acts windup as well.

Windup can be prevented by ramping up to the setpoint slowly preventing buildup of integral error[**astrom2010feedback**] . This can be considered for future versions especially for depth control.

4.1.3 Derivative Constant

The derivative constant acts on the rate of change of the response. This in turn determines the stability of the system and helps cut down oscillations.

4.1.4 Tuning and Real World Use

On Bumblebee AUV all degrees of freedom except the depth controller usually do not require integral control and work on just proportional and derivative terms. This differs from industrial statistics in which a proportional integral controller is preferred.

The tuning of the controller is done by hand following the Ziegler-Nichols method[**ziegler1942optimum**][**astrom2010feedback**]

1. Turn off K_d and K_i
2. Increase K_p till the controller can achieve the setpoint within a reasonable period of time
3. Observe the overshoot and oscillations and introduce K_d to correct them. Note that K_d will often also cause oscillations when used in excess.
4. If there is a clear steady state offset between the setpoint and the input, and the controller is not oscillating or even attaining the setpoint, introduce K_i

4.1.5 Tuning UI

A graphical user interface to tune the above developed PID controller was developed. The UI uses `qcustomplot` together with Qt 5.6 (C++) to achieve simultaneous high frequency plotting of all 6 dof in the same window. Having all the control loops in the same window is useful as tuning one loop may introduce disturbances to another, particularly because multiple thrusters are sharing certain degrees of freedom.

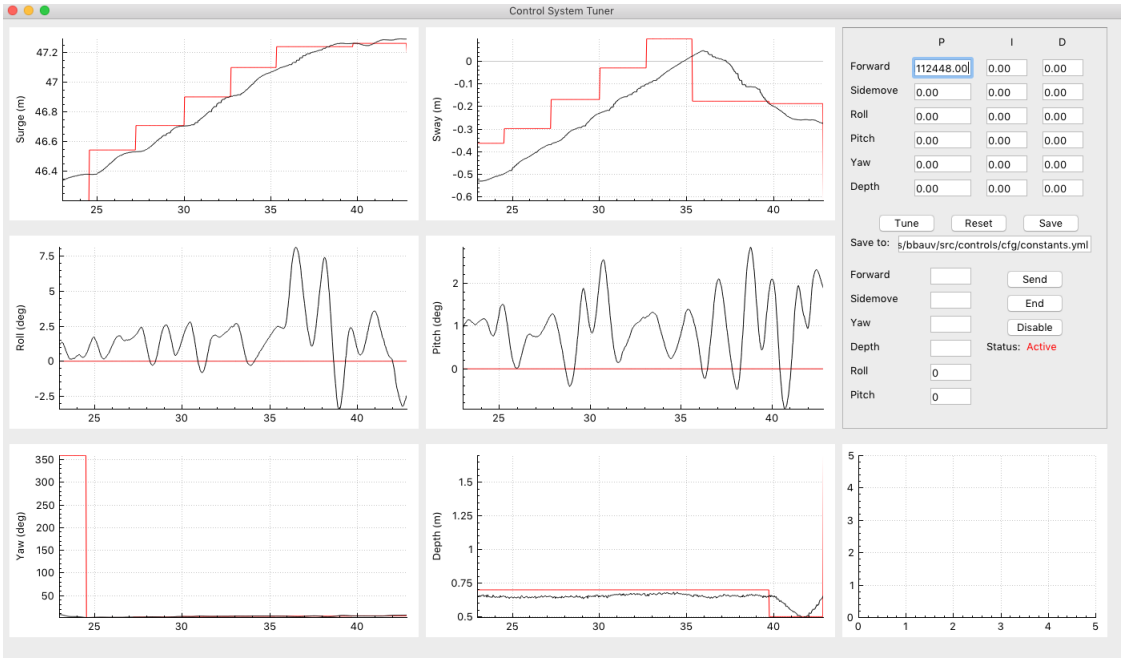

Figure 4.1: Control Systems Tuning UI

The tuning UI is shown in 4.1 and adapted to both the AUV and the ASV.

4.2 Adaptive PID Controller

Closed loop PID controllers work well in real world conditions even when we have no system identification or modelling done[**astrom2010feedback**] . It requires tuning of parameters by hand and is described in the section above. An AUV running a hand tuned PID system has worked well for us in pool environments.

On the surface craft however, the PID controller performed badly every time there was a significant environmental disturbance such as the wind. Underwater, this would correspond to currents albeit in a very simplified manner. Currents are more homogeneous and the AUV should be able to keep up given power enough actuators.

In this section we will talk about an implementation of an adaptive PID controller using a simple neural network.

4.2.1 Feature Extraction

Equation (4.1) shows the output of a PID controller as a function of the error and the three constants. As such, we will try to extract three features from the resultant signal that indicate the quality of the three PID constants.

Rise Time

The rise time of the PID controller is the time t that was taken for the sensor reading to have changed from the original value to when it has achieved the set point. We only consider the raw time t taken till the set point crosses

and not time to settle.

Periodicity

The process oscillation period with respect to gain used can be identified using the discrete Fourier Transform. We first take n samples of the response from the closed loop controller and compute the Fourier transform

$$F(x) = \frac{1}{n}\sum_{t=0}^{n-1} s(n)e^{-i2\pi\frac{kn}{n}}$$

Even for most real valued functions, the Fourier transform is complex and thus $F(x)$ is composed as

$$F(x) \sim \Re(x) + \Im(x)$$

which can be used directly as the feature set without extracting the bin with the highest number of elements. The feature set is thus

$$\{\Re(x), \Im(x)\}; \quad x = 0 \ldots n$$

Typical oscillations seen from the AUVs feedback control system are of low frequency. After we perform the FFT on the input we discard the high frequency components. This feature set, without much modifications has been used in the past for the same purpose[**Swiniarski1990**]

Steady State Error

The steady state error is a feature that can be calculated only after the controller has reached its setpoint. It is time delayed for the oscillations to

settle (if it does) and may often be 0. The feature is collected even when the integral term is turned off for training.

$$E_s = s - \frac{1}{N}\sum s(n)$$

4.2.2 Neural Network Tuner

I used a static feed-forward and back propagated neural network with a sigmoid activation function. The neural network was implemented in Python using the `keras` library. The number of input neurons was set the size of the feature vector (in this case each FFT bin was considered an independent feature set plus the rise time). Steady state error feature vector was not used in this experiment. The loss function was set to mean squared error.

The learning set that was developed for this training consisted of

$$V = \{F, T_s\}$$

where F is the periodicity feature vector and with labels

$$L = \{K_p, T_i, T_d\}$$

and only one degree of freedom was used, depth control. Depth control was chosen because it had all 3 PID terms enabled and showed minor oscillations due to physical constraints even when tuned extensively. The model trained is shown in 4.2

The loss (mean square error) over 100 epochs of training is shown below in figure 4.3

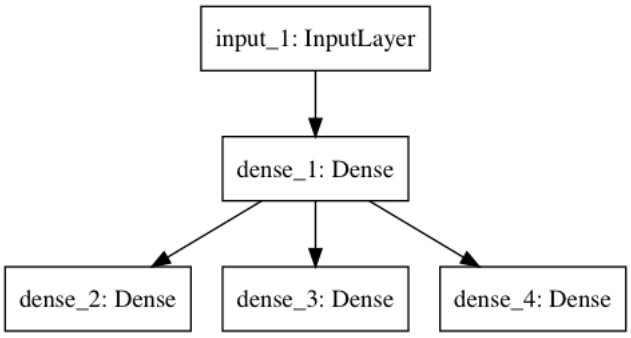

Figure 4.2: Neural Network Model

Even though from fig 4.3 it seems that the network is learning, the output from the network is unstable. The network attempts modification of stable PID parameters when no intervention is necessary (steady state). The highly non deterministic nature of the learned model makes it hard for a human to tune the PID to their preferences after it has been deployed.

On the other hand, instead of regression, the use of a classifier is possible to detect windy conditions. In this scenario the controller will be hand tuned in many different weather conditions and the set of optimal parameters would be chosen. The system then fits the PID behaviour against known labels of weather conditions. The performance of the PID is then continuously monitored and the system switches parameters automatically based on detected weather.

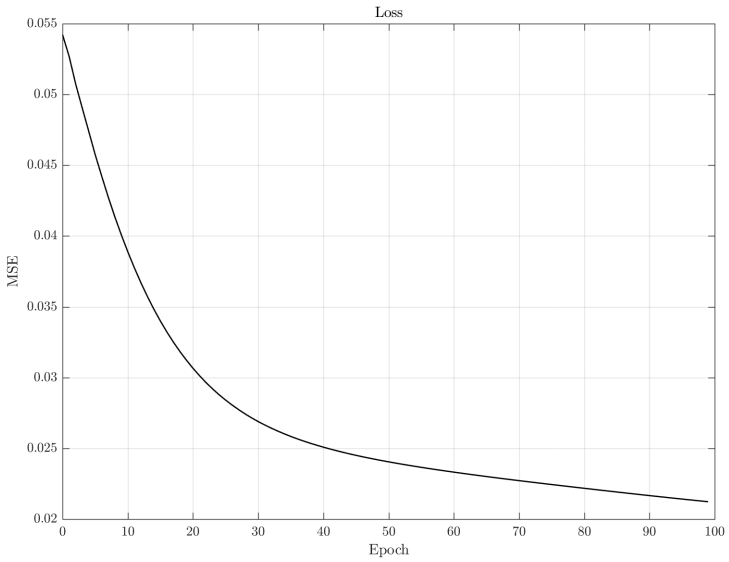

Figure 4.3: Model training loss (MSE)

Chapter 5

Navigation

5.1 Filter Design and Architecture

The underwater vehicle and the surface craft have slightly different navigational requirements. The surface craft carries an onboard GPS receiver which provides global navigation while the AUV does not have any absolute frame sensors, only relative. As such the requirements for the navigation solution on board the AUV is given as

- Relative navigation (referenced to start position)
- Able to compensate for lack of DVL bottom track for upto 30 seconds
- Absolute positioning error over 10 minutes of operation must be less than 0.5m

while for the surface craft it is

- Local relative frame and its corresponding global frame navigation

- Resistant to heavier local perturbations (swells, wind)
- Highly responsive

The GPS aided navigation for the ASV does not compensate for GPS outages over reasonable expectations[1] as the ASV operates in open sea with no tall structures to obscure the signals. Multipath effects still need to be accounted for however, from other structures on the ASV and the surface of the water.

The requirements for each vehicle is considered and merged into a single filter. In the future the AUV will be equipped with an USBL sensor (Ultra Short BaseLine) which measures the distance and azimuth from a transponder mounted on the ASV to the AUV giving us an offset from the ASV. This helps us significantly in observing the positioning bias accumulated by the DVL on the vehicle.

5.2 Comparison between IMUs

5.2.1 STIM300

The STIM300 inertial measurement unit is a tactical grade IMU that comes with an accelerometer, gyroscope and inclinometer. The accelerometer sensitivity of the instrument can be configured and we used 5g for a lower noise floor (over the 10g) and better sensitivity. It is unlikely to encounter high jerk in an underwater environment which is already damped. Unlike a traditional

[1]Not more than 1 minute

IMU, the STIM300 does not come with a magnetometer. The advantages are

- Not affected by magnetic fields and giant ferromagnetic structures such as ships, around which an AUV is likely to operatre
- Inclinometer offers better sensitivity to orientation changes (tilt) at $2G$ compared to the main accelerometer

The measured sensor mean and standard deviation when kept stable is shown in table 5.1

The sensor shows high turn on static bias and must be calibrated for before use in any filters. The standard deviation of the sensor readings is within bounds according to the datasheet.

	Measured μ	Measured σ
Gyroscope std dev (rad)	-0.000765	0.000258
Accelerometer std dev (ms^{-2})	0.552	0.0026
Inclinometer std dev (ms^{-2})	0.551	0.0075

Table 5.1: STIM300 measured sensor characteristics

5.2.2 Sparton AHRS-8

The Sparton AHRS-8 is an inertial measurement unit that contains gyroscopes, accelerometers and magnetometers. The advantage that is offers us is that the unit contains a magnetometer which makes z axis bias observable.

The magnetometer is however noisy and needs calibration against hard and soft iron biases.

	Measured Mean	Measured std dev
Gyroscope std dev (rad)	-0.071	0.0031
Accelerometer std dev (ms$\hat{2}$)	-0.077	0.0086
Magnetometer std dev (milliguass)	0.03	0.047

Table 5.2: AHRS8 measured sensor characteristics

5.2.3 SBG Ellipse

The SBG Ellipse which is on loan seems to be having some firmware bugs. While it does output the GPS sattelite fix, its internal EKF seems to have crashed or does not update. The unit will be replaced and measurements will be taken again with a new sensor.

5.3 Comparison between GPS Antennae

In this section we compare the results obtained in stationary position between two different GPS antennae connected to the same GPS reciever. Both the antennae are passive and is powered by an low noise amplifier (LNA) on the receiver board. The two antennae used are the ANTCOM-G5AntA1T1 and ANTCOM-G5Ant53A4T1 and the characteristics are shown in table 5.3

As can be seen on figures 5.1 and 5.2 ANTCOM-53A4T1 performed better showing reduced standard deviation in its measurements. It should be noted

that the antenna (53A4T1) showed a lower standard deviation in spite of using less satellites (11) as opposed to A1T1 which used 15.

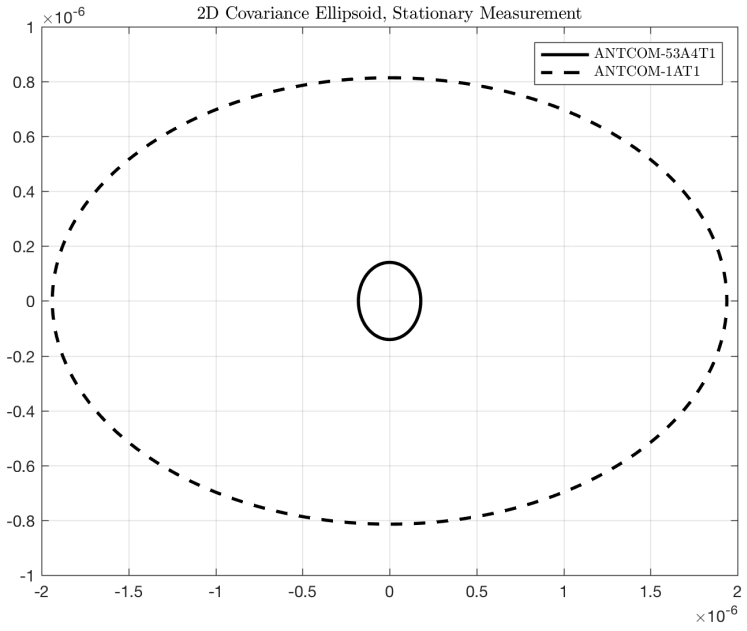

Figure 5.1: 2D covariance ellipsoid comparison, x, y in degrees

It is recommended to use ANTCOM-G5Ant53A4T1 for projects requiring precise positioning.

Antenna	G_1	G_2	Std Dev (longitude)	Satt Used vs Satt Available
53A4T1	-2.1	-2.0	1.19×10^{-7}	11/15
A1T1	-4.1	-3.9	6.9×10^{-7}	15/15

Table 5.3: Antenna characteristics comparison where G_1 is antenna gain at 20 deg elevation, GPS L1 and G_2 is gain at 20 deg elevation for GPS L2, dBic

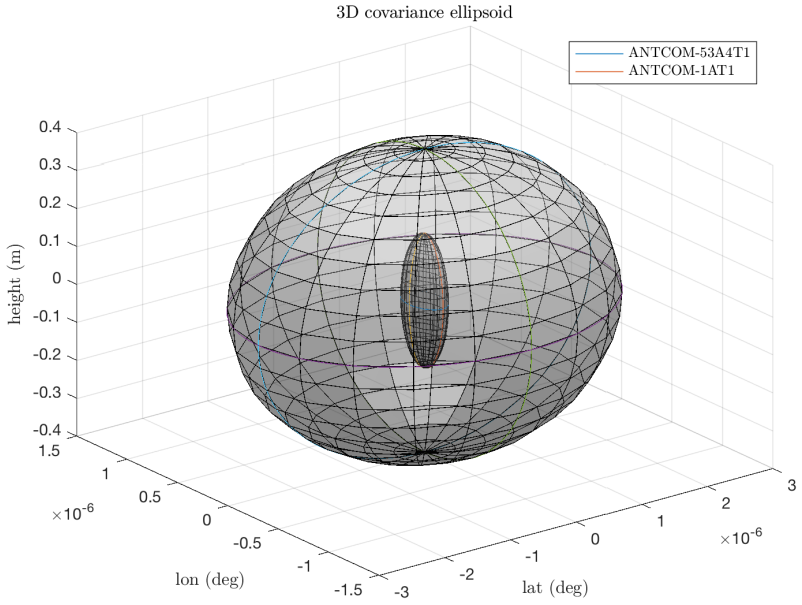

Figure 5.2: 3D covariance ellipsoid comparison. The inner ellipsoid is by 53A4T1.

It is speculated that the antenna with the bigger dish performs better because of its bigger ground plane giving it an advantage in SNR. The bigger dish also has a better circular polarization gain at a 20 degree angle around, in which there are lots of GPS satellites.

5.4 Preprocessing GPS Data

GPS data is received from the board in the world frame LLA coordinates as well as world frame velocity and their associated standard deviations. The first few seconds of a good GPS (with respect to number of satellites tracked and the standard deviation) is used to set an origin. The WGS84 geodetic

system is used.

Working in local frame has many advantages

- The AUV operates in pure dead reckoning mode most of the time due to lack of GPS or USBL hence without the world transform
- The error terms in the Kalman filter are smaller
- Both the velocity and the positional fixes from the GPS provide observability to yaw axis drift equally
- The control system operating in the local frame does not need to be transformed

In addition to offsetting the origin, the magnetic field declination at the precise GPS LLA coordinate is calculated dynamically and applied to the quaternion attitude.

5.5 Kalman Filtering

Usual techniques for navigational sensor fusion include the Extended Kalman Filter (EKF), Error State Kalman Filter (ESKF) and the Unscented Kalman Filter (UKF). Each sensor fusion method has its own pros and cons for use in different environments. In previously done work I implemented an ESKF for use with the AUV. The error state kalman filter is particularly robust in environments which are highly non linear and hard to model or does not fit

the assumptions of the full state EKF. My previous motivations for choosing the ESKF were

- Error state always operates close to the origin and protects from singularities and guarantees linearlization
- The error state is always minimal in magnitude hence we can discard second order products making computation of Jacobians fast

More benefits of the ESKF can be found in [**madyastha2011extended**]

Originally developed for use onboard the ASV and now iterated on and used for comparison we also have an Unscented Kalman Filter. The advantages the UKF has over a traditional EKF are

- Derivative free - does not require computation of Jacobian matrices
- Avoids errors rising from linearization in traditional EKF using the unscented transform

According to [**Wan2000**] the UKF does not impose a higher computational load when compared to the EKF. It requires more computation and memory space when compared to the ESKF.

5.5.1 Kinematic Equations

The kinematic equations that is used in either filter design is as follows

$$p = p + v\Delta t + \frac{1}{2}(R(a) + g)\Delta t^2 \tag{5.1}$$

$$v = v + (R(a) + g)\Delta t \tag{5.2}$$

$$q = q \otimes q\{\omega \Delta t\} \tag{5.3}$$

$$a_b = a_b \tag{5.4}$$

$$w_b = w_b \tag{5.5}$$

where R is the rotation matrix obtained from the quaternion orientation and $q\{x\}$ is the quaternion composed from the rotation x formed by setting the real part of the 4 vector to 0 and the imaginary part to the vector. p, v, a_b, w_b, g are vectors corresponding to the position in meters, velocity in ms^{-1}, accelerometer bias, gyro bias and the gravity vector $\begin{bmatrix} 0 & 0 & g \end{bmatrix}^T$ respectfully. The gravity vector is parameterised as it can vary by location.

We can then form the kinematic state model by taking the derivative of each with respect to time.

5.5.2 Error State Kalman Filter

We define the error state vector δx as

$$\delta x = \begin{bmatrix} \delta p \\ \delta v \\ \delta \theta \\ \delta a_b \\ \delta w_b \end{bmatrix} \tag{5.6}$$

where the true state $x = x + \delta x$. Its dynamics follow from the kinematic equations:

$$\delta p = \delta p + \delta v \Delta t \tag{5.7}$$

$$\delta v = \delta v + (-R([a]_\times \delta\theta) - R\delta a_b + g)\Delta t \tag{5.8}$$

$$\delta q = R^T[\omega]_\times \delta\theta - \delta w_b \Delta t \tag{5.9}$$

$$\delta a_b = \delta a_b \tag{5.10}$$

$$\delta w_b = \delta w_b \tag{5.11}$$

which we then represent as

$$\delta x + i$$

where i holds the random impulses applied to the state.

The state transition matrix in continuous form A can be obtained from (5.7)

The random impulse matrix Q is obtained by measuring the standard deviations of the sensors in stationary state

$$Q = \begin{bmatrix} \sigma_p^2 & & & \\ & \sigma_v^2 & & \\ & & \sigma_a^2 & \\ & & & \sigma_\omega^2 \end{bmatrix} \tag{5.12}$$

$$A = \begin{bmatrix} 0 & I & 0 & 0 & 0 \\ 0 & 0 & -R[a]_\times & -R & 0 \\ 0 & 0 & -[\omega]_\times & 0 & -I \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.13}$$

Which can then be discretized to time step Δt with a second order taylor approximation as

$$F_d = I + A\Delta t + \frac{1}{2}A^2\Delta t^2 \tag{5.14}$$

Prediction

The error state is predicted as

$$\delta x = F\delta x \tag{5.15}$$

but since the error state is initialized to 0 it always returns 0. The covariance is propagated forward

$$P = FPF^T + Q \tag{5.16}$$

this equation shows us that with every prediction step the covariance grows bigger (due to the addition of the noise term in Q).

Correction

Now that we have the predicted error state we wait for sensor readings to arrive. The sensor readings must be relatable to the true state vector from

which we can find the residual or the measured error state.

Example: Arrival of DVL velocity data

We take the measurement vector

$$y = \begin{bmatrix} 0 \\ v_m \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

and the measurement matrix H as

$$H = \begin{bmatrix} 0 & & \\ & R^T & \\ & & \ddots \end{bmatrix}$$

to obtain the residual r as

$$r = y - Hx \tag{5.17}$$

The measurement has its own covariance which we denote by matrix N. The Kalman gain is defined as

$$K = PH^T(HPH^T + N)^{-1} \tag{5.18}$$

Rewriting and taking the limit,

$$K = \lim_{P \to 0} \frac{PH^T}{HPH^T + N} = 0 \tag{5.19}$$

we can see that when our process model is accurate the process covariance will be small which leads to a small to insignificant gain on the measurement.

This means that the filter will prefer its predictions over the measurements. It can also be seen that when the measurement noise N is high the filters gain will be low forcing the filter to rely on its own prediction. This can be a problem if the process covariance is itself high at this point. [**Wan2000**]

To solve for K we can rewrite (5.18) as

$$K = PH^T(HPH^T + N)^{-1} \tag{5.20}$$

$$K(HPH^T + N) = PH^T \tag{5.21}$$

$$(HPH^T + N)^T K^T = HP^T \tag{5.22}$$

which is of the form $AX = B$

A covariance matrix is guaranteed to be positive semi definite as a covariance between two random variables is defined as

$$\sigma_{x,y} = E[(x - E(x))(y - E(y))]$$

and is invariant to positional changes giving us a symmetric matrix. Diagonalization of the matrix gives us a diagonal matrix containing variances which can only be 0 or positive.

The solution for K can then be found with a robust Cholesky decomposition.

The error state is then defined as

$$\delta x = Kr \tag{5.23}$$

and is ready for injection to the true state for correction. The filter covariance P needs to be updated as[**Wan2000**]

$$P = (I - KH)P(I - KH)^T + KNK^T \tag{5.24}$$

5.6 GPS Fault Detection

Most GPS receiver boards are not capable of filtering out multipath effects without additional sensors. GPS multipath is when the receiver wrongly reads a reflected signal instead of the shorted path. This can cause significant jumps in GPS positioning. A chi square test is usually employed to check the hypothesis that given our past sampling of measurements which lie in a certain distribution, is the new measurement vector realistic[**Chambers2014**].

The chi square statistic is computed by simply taking the innovation covariance from the update step.

$$\chi^2 = z^T S^{-1} z$$

where S is from (5.18)

$$S = (HPH^T + N)$$

The chi square table can then be consulted for the required confidence and with the available degrees of freedom to make a prediction of it is a completely wrong measurement. Chi square fault tolerance does not take up significant computing resources and is easy to implement.

5.7 Results

A test jig was designed on a trolley. All 3 different IMUs and GPS antennas were mounted to the cart. The test location that was used was the NUS track. The outer ring of the track served as a rough guide.

A track generated by pushing the trolley around the field is shown in 5.3. `rviz` with a tile from MapBox overlayed was used to generated the visualization. However, the tile is not very high quality and is offset slightly. The ESKF is run at 100Hz. Near the top left of the track there was a 1.57 minute GPS outage. As discussed in the requirements earlier, since we are going to be operating at sea with an entire sky dome visible at all times, extra long GPS outtages were not tuned or accounted for.

The filter works well in filtering unwanted jumps in position and velocity as can be seen from figure 5.5. These jumps happen when the receiver reacquires fixes to various satellites or loses a fix.

The steady state performance (fig 5.6) of the filter shows good rejection against unwanted noise while being highly responsively to small jerks that can affect the control systems.

	Standard deviation	Expected drift
Position (GPS)	0.07805m	
Position (DVL)	0.01795m	2.1m/100m
Orientation	0.001541deg	
Heading		1.7 deg/hr

Table 5.4: INS System Measurements

The measurements given simulated runs against bagged ROS data is given below 5.4. GPS standard deviation is acceptable and so is the orientation estimation. DVL only drift over an hour is higher than expected, but because it is fully relative, the heading drift affects the positioning and rehoming back significantly.

Figure 5.3: Field test - filtered GPS track

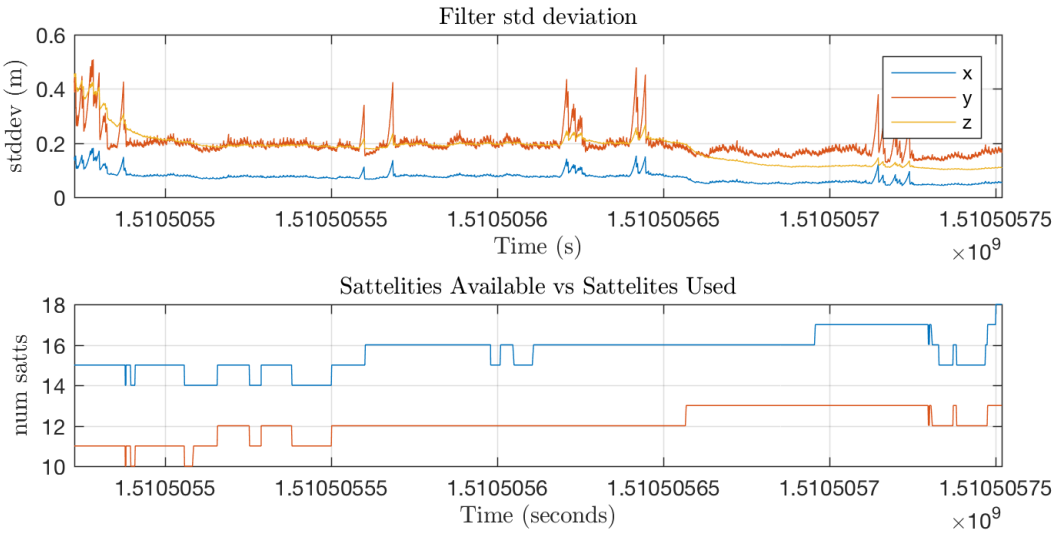

Figure 5.4: Rejection of sudden jumps

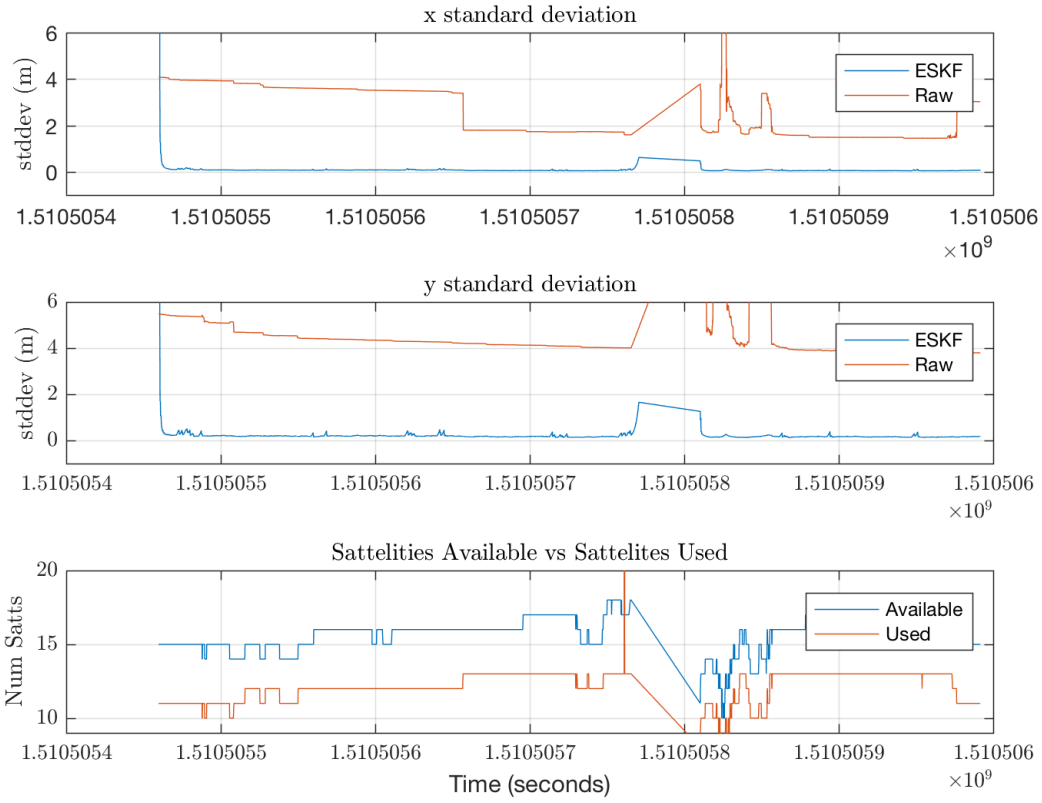

Figure 5.5: Effectiveness

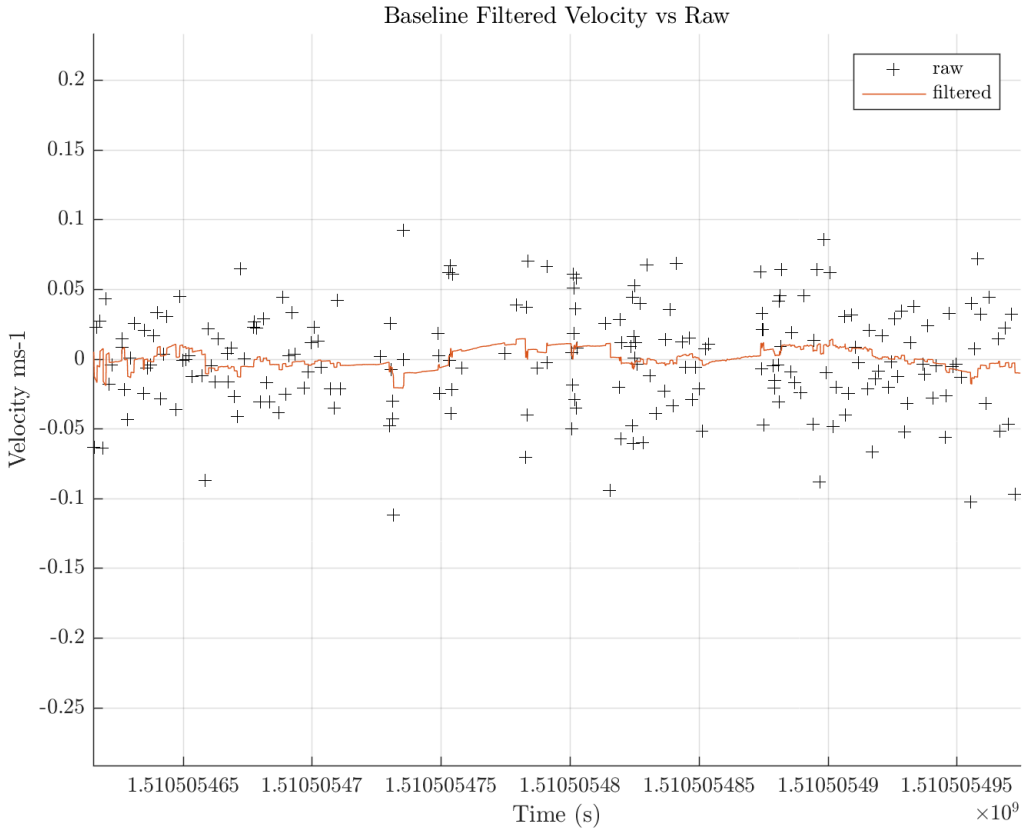

Figure 5.6: Baseline Filtering (not moving)

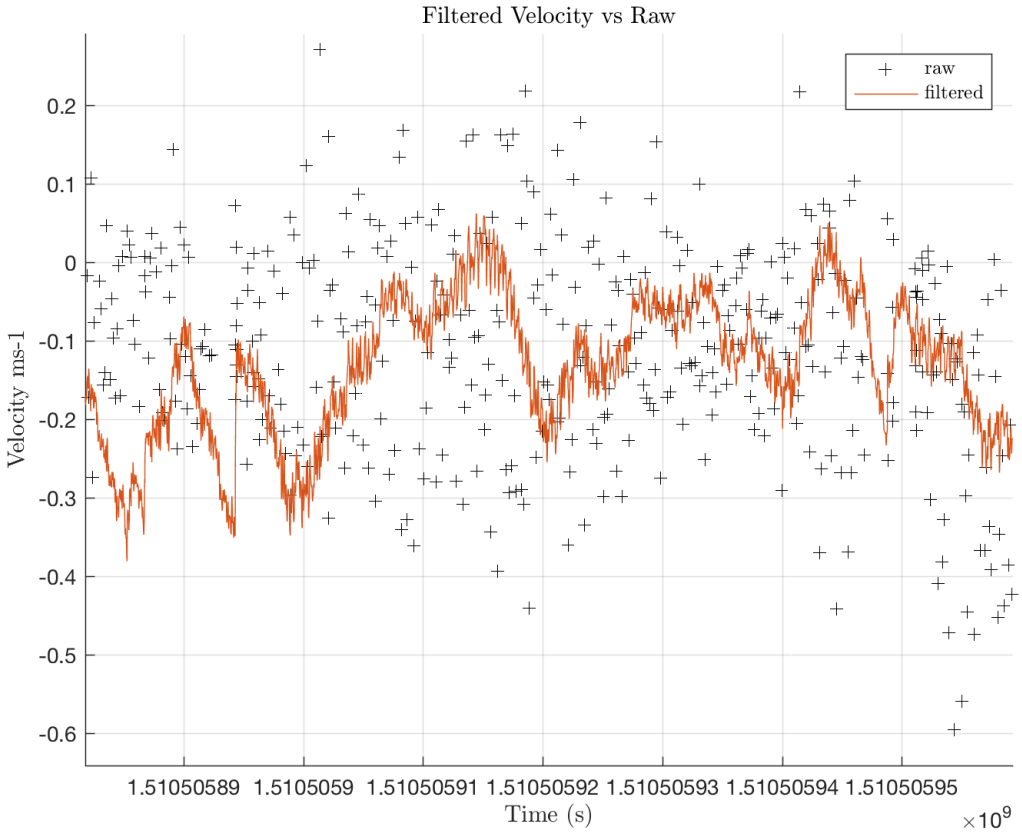

Figure 5.7: Velocity Filtering (moving)

Chapter 6

Conclusion

An underwater simulation package was maintained and new inertial sensors added which is essential for simulation of an AUV and USV. Hydro dynamic models of the AUV was developed and use in the simulator together with its existing control systems. Thrust of each of the AUV thrusters was measured for use in both simulation and tuning of control systems. The simulator was used to tune the control system and generate feature vectors to train a new neural network based adaptive parameter estimator. Even though the results from the parameter estimation was not good, an alternative concept whereby we tune and let the system dynamically switch profiles based on weather conditions was developed.

GPS, its basics and principles were studied and implemented. Multiple GPS antennae were tested and and suggested. One of the biggest issues we had at RobotX 2016 was the GPS receiver not working properly as well as not having a full fledged, GPS aided INS system. The INS systems were

combined to one package which can be run on either system or only one. The GPS system in absence of a boat was taken field testing multiple times to tune and develop new methods. Even though not detailed an Unscented Kalman Filter was developed to compare, and no major differences was found - both vehicles have been standardised to the same system.

Some drawbacks of the new system include slightly more than expected jumps from the GPS receiver reacquiring fixes and I had a lot of problems with magnetometer calibration. In the next iteration of the filter a tightly coupled approach will be trialled as it has been proven to be accurate with as less as 3 satellites for a short period of time. New field tests will be conducted with a LIDAR providing mapping features so that real world drift and inaccuracies can be measured numerically. In addition to that, Time of Validity tests need to be performed in an unstable environment.