NEURAL NETWORK MODELING OF THE DYNAMICS OF AUTONOMOUS UNDERWATER VEHICLES FOR KALMAN FILTERING AND IMPROVED LOCALIZATION

BY

SHARAN BALASUBRAMANIAN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Industrial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Adviser:

Clinical Associate Professor William Robert Norris

ABSTRACT

Autonomous Underwater Vehicles (AUVs) and Remotely Operated Vehicles are used for a variety of underwater operations and deep-sea explorations. One of the major challenges faced by these vehicles is localization i.e., the ability these vehicles to identify their location with respect to a reference point. The kinematic Extended Kalman filters have been used in localization in a method known as dead reckoning. The accuracy of the localization systems can be improved if a dynamic model is used instead of the kinematic model. The previously derived dynamic model was implemented in real time in UUVSim, a simulation environment. The dynamic model was tested against the kinematic model on various test courses and it was found that the dynamic model was more stable and accurate than the kinematic model. One of the major drawbacks of the dynamic model was that it required the use of numerous coefficients. The process of determining these coefficients was extensive, requiring significant experimentation time. This research explores the use of a Neural Network architecture to replace these dynamic equations. Initial experiments have showed promising results for the Neural Network although modifications will be required before the controller can be made universally applicable.

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr. William Robert (Bob) Norris for his invaluable support during my degree. His guidance helped me to grow both professionally and personally in the last two years. I would like to thank Dr. Michael Dyer from the department of Finance for taking me in as his Teaching Assistant and providing me with an opportunity to teach for the last two semesters. I would also like to thank Dr. Richard Sowers and Dr. Ramavarapu RS Sreenivas from the department of Industrial and Systems Engineering for their advice.

I would like to thank my parents for providing the necessary moral and financial support for pursuing a Master of Science degree. I would also like to thank my roommate and my friend Mr. Gnanendra Reddy for putting up with me and taking care of me during tough times.

My thesis research would not be possible without the contribution of my project group, nicknamed the "Submarine Bois". Rodra Wikan Hascaryo, one of the predecessors of this research played a huge role by my side to help me understand and design the dynamic model. The support provided by Sandeep Srikonda and Ayush Rajput during the implementation phase of the Kalman Filter cannot be overstated. Two hard working undergraduate students, Chirag Rastogi and Justin Yurkanin played a vital role in the data collection phase.

Finally, I would like to acknowledge the members of the Autonomous and Unmanned Vehicle Systems Laboratory and my friends for their help and motivation during the last two years.

This research has been funded through a generous gift grant from Schilling Robotics, a subsidiary of Technip FMC.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

The planet Earth consists of 71% water and 29% land. Recent scientific advancements have enabled us to put satellites on space to monitor various events on Earth. These geo-spatial satellites can provide us information about the surface temperature of oceans, the direction of ocean currents, color of ocean bodies etc. But what these satellites lack is their ability to look deeper into the oceans in 3D. According to Gene Feldman, we have more knowledge about the surface of Mars than that of our ocean bodies [1]. At deeper depths, the pressure exerted by these cold and dark waters is enormous. Luckily, we have Autonomous Underwater Vehicles (AUVs) and Remotely Operated Vehicles (ROVs) for underwater and deep-sea explorations.

The most basic type of ROV is the RexROV and it is based on the SF 30k ROV [2]. It has four lateral thrusters and four vertical thrusters for maneuvering purposes. The four lateral thrusters are responsible for its movement along the x and y axis. The four vertical thrusters ensure that the vehicle remains floating when it is stationary and are responsible for movement along the z-axis. The RexROV with the thrusters is shown below.

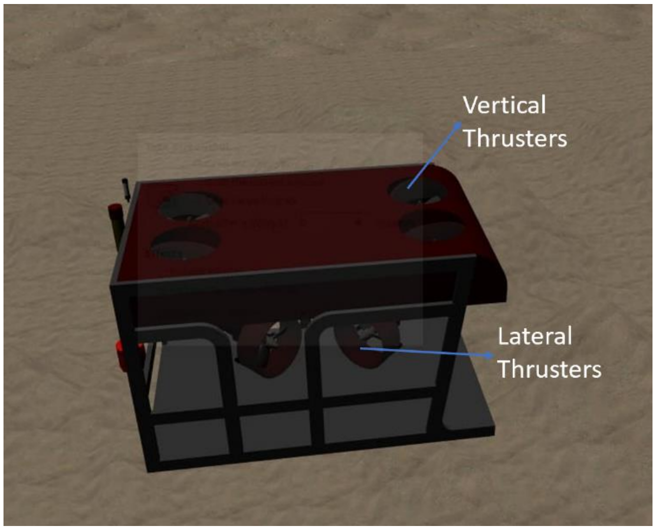

Figure 1: RexROV with thrusters

Experimentation with an AUV was difficult since it was hard to control the nature of the environment in which they were operated. A realistic simulation will greatly help in reducing the risks and costs involved in validating several aspects of a project. There are two major simulator environments used in reference to underwater robotics. One is UWSim [2] and the other is

UUVSim. The UWSim is a kinematic simulator which is primarily used as a visualization tool [3]. UWSim has been used as a part of the Trident project and it renders high quality graphics with the help of OpenSceneGraph (OSG) and osgOcean. The UUVSim package contains the implementation of Gazebo plugins and ROS nodes necessary for the simulation of unmanned underwater vehicles. The UUVSim plugin consists of several sensors and allows for simulating the hydrodynamics of an AUV. For our purposes, UUVSim has the most applicable packages and plugins required to simulate the underwater environment. Gazebo runs the UUVSim plugin as shown below.

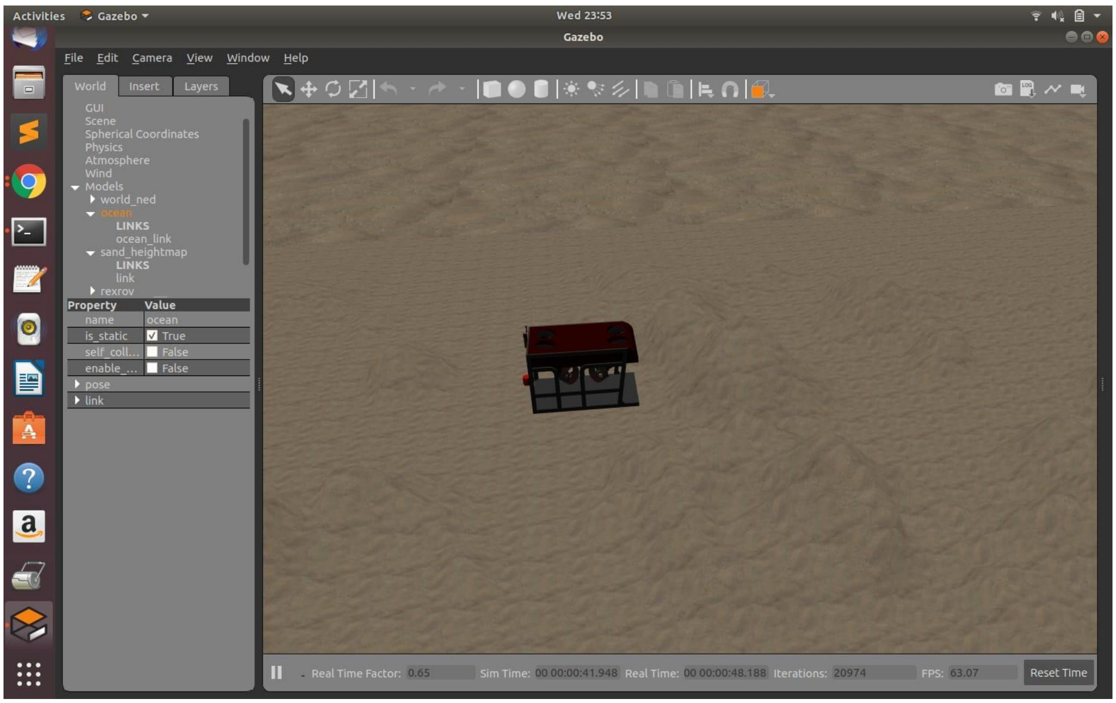

Figure 2 : UUVSim running on Gazebo

One of the key aspects in operating an autonomous vehicle is knowing its position with respect to the environment. The process by which a vehicle can figure out where it is in the environment is called localization and there are several methods applicable to the problem. One of the most prominent localization techniques is using a Kalman filters [4]. The Kalman filter uses a system model to predict the position of the vehicle. Romagos used a kinematic model of an underwater vehicle to predict the system states [6]. The kinematic model deals with the motion of the vehicle without taking the forces acting on the vehicle into account. It has been suggested the use of a dynamic model instead of a kinematic model would yield better results in terms of accuracy, since it considers the forces acting on the vehicle. Hascaryo has provided a detailed derivation of the dynamic model and provided the preliminary results in his thesis [7]. This research involves improving the work done by Hascaryo with the use of neural networks as an alternative for modeling the dynamics of the underwater vehicle.

1.2 METHODOLOGY

Based on the work of Fossen, a dynamic model which has been observed to the *de facto* standard has been implemented [2][8][9][10]. First, a generic 6 degrees of freedom (DoF) dynamic model for an underwater vehicle was developed and then simplified into a 4 DoF model for RexROV. 2 DoF were eliminated due to the inherent stability of the vehicle which helped in reducing the 6x6 matrix to a pseudo 4x4 matrix thereby improving computing power. The 4 DoF model incorporating the values derived by Berg [8], was used in the Extended Kalman filter (EKF) equations and the output was used as input to a pure pursuit controller. The output of the dynamic model was compared to the sensor values for verification. The pure pursuit controller provided the steering angle required to drive the RexROV in a Robotic Operating System (ROS) Gazebo based simulation environment, UUVSim [2]. Three behavior evaluation test courses, as suggested by Norris [11], were used to evaluate the performance of kinematic and dynamic model driven controllers.

The dynamic model required several coefficients, whose estimation can be difficult for real time operations. A Neural Network based architecture in Tensor Flow [12] and Pytorch [13] was developed to emulate the performance of the dynamic model. The fully connected network was used to calculate the dynamic model values and fed to the Kalman filter and its performance was compared with the dynamic model. The architecture was experimental, and several modifications were made to improve the performance of the neural network based on the input parameters.

1.3 THESIS ORGANIZATION

This thesis was organized to demonstrate the performance of the dynamic model and how Neural Networks can be used to overcome the drawbacks of a dynamic model. The introduction chapter presents the motivation behind the research and the methodology. The research was conducted on RexROV, a type of submarine, in a Gazebo based simulator called UUVSim.

Chapter 2 outlines the derivation of a generic dynamic model for an underwater vehicle. The derivation was based on the *“The Handbook of Marine Craft and Hydrodynamics”* by Fossen.

The dynamic model was simplified for a RexROV and it was implemented in UUVSim. The results of the dynamic model were compared with the IMU sensor values for evaluation.

Chapter 3 provides a brief introduction to the Kalman filter algorithm and the results of the performance comparison between the dynamic model and the kinematic model on various test courses using a pure pursuit controller. Chapter 4 demonstrates various Neural network architectures that were used to model the dynamics of RexROV and their performance on various test courses. Lastly the conclusion was provided in chapter 5. Future work and further modifications to improve the performance of Neural Networks were suggested.

CHAPTER 2: DYNAMIC MODEL DERIVATION FOR RexROV

2.1 COORDINATE SYSTEMS AND CONVENTIONS

As discussed in the previous chapters, this research used the UUVSim based Gazebo simulation environment. The simulation environment consisted of two bodies – the environment (here it was the sea) and the vehicle. The environment was in North-East-Down (NED) frame [14] while the vehicle was in BODY frame which follows the Society of Naval Architects and Marine Engineers (SNAME) 1950 convention [14]. The BODY axes x, y and z and SNAME 1950 convention for RexROV are shown in figure 3.

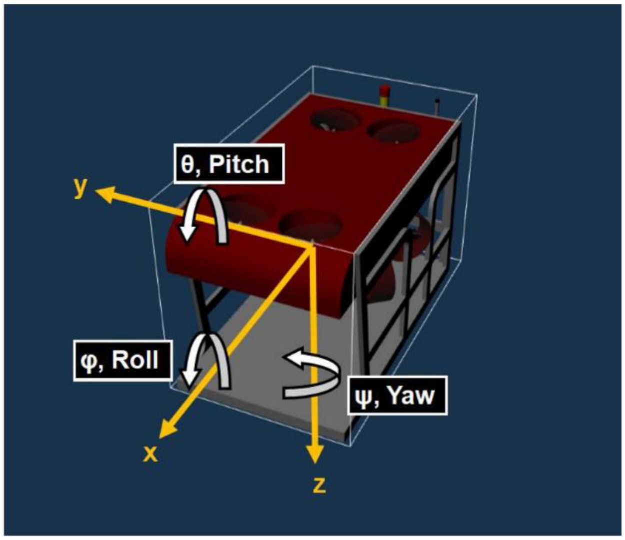

Figure 3: RexROV SNAME convention

The terms used in the 1950 SNAME convention are listed in table 1.

Table 1 : SNAME Convention for marine vehicle

Term	Position/Euler Angle	Velocity	Forces and Moments
Surge	x	u	X
Sway	y	v	Y
Heave	z	w	Z
Roll	φ	p	K
Pitch	θ	q	M
Yaw	ψ	r	N

The transformation from BODY to NED coordinates is done using the following equation.

$$\eta_{b/n}^{\dot{n}} = J_b^n(\Theta) v_{b/n}^b \tag{1}$$

Here, $\eta_{b/n}^{\dot{n}}$ represents the transformed velocity variables in NED coordinates, $v_{b/n}^b$ are the linear and angular velocities of the submarine in BODY coordinates and $J_b^n(\Theta)$ represents the transformation matrix. The transformation matrix, $J_b^n(\Theta)$ was given by,

$$J_b^n(\Theta) = \begin{bmatrix} R_b^n(\Theta) & 0_{3x3} \\ 0_{3x3} & T_b^n(\Theta) \end{bmatrix} \tag{2}$$

In the above matrix, $R_b^n(\Theta)$ represents a 3x3 rotation matrix for the linear velocities, $T_b^n(\Theta)$ represents the 3x3 rotation for the angular velocities and 0_{3x3} represents a 3x3 zero-matrix. The $R_b^n(\Theta)$ and $T_b^n(\Theta)$ matrices were calculated as shown below. For simplicity, angle A's cosine, sine and tangent values were represented as cA, sA and tA, respectively. The underlying condition for these rotation matrices was cosA≠0 i.e, A was not $\frac{\pi}{2}$ or $\frac{3\pi}{2}$.

$$R_b^n(\Theta) = \begin{bmatrix} c\psi c\theta & s\phi s\theta c\psi - s\psi c\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & s\theta s\psi c\phi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\theta c\psi \end{bmatrix} \tag{3}$$

$$T_b^n(\Theta) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{4}$$

2.2 COMPONENTS OF THE DYNAMIC MODEL

The forces acting on an underwater vehicle can be classified into five categories namely,

- Kinetic forces
- Hydrodynamic forces
- Hydrostatic forces
- Actuator forces
- Disturbances

The 6 Degrees of Freedom (DoF) model for an underwater vehicle can be represented as follows [14].

$$\dot{\eta} = J(\eta) v \tag{5}$$

All of these forces were modeled separately and then added together to determine the dynamics of the underwater vehicle.

$$M\dot{v} + C(v)v + D(v)v + g(\eta) + g_0 = \tau + \tau_{wind} + \tau_{wave} \tag{6}$$

The terms used in the equation were:

η - Vehicle position and pose

v - Vehicle velocity

J - Transformation matrix between NED and BODY coordinate systems

M - inertial matrix

C(v) - Coriolis forces

D(v) - Damping matrix

g(η) - Buoyancy force

τ, τ_{wind} *and* τ_{wave} – Actuator, Wind and Wave forces respectively.

2.2.1 KINETIC FORCES

The underwater vehicle was considered as a rigid body and it does not undergo any deformation during its operation. For the purpose of simplicity, it was assumed that the center of mass coincided with the center of gravity (CG). The kinetic force was the force required to move the submarine from its state of rest. The kinetic force (τ_{RB}) consisted of the inertial component (M_{RB}) and Coriolis component ($C_{RB}(v)$) as shown. Here the subscript RB represents rigid body.

$$\tau_{RB} = M_{RB}\dot{v} + C_{RB}(v)\mathrm{v} \tag{7}$$

Here v and $\dot{v}$ represent the velocity and acceleration vectors. The velocity and acceleration vectors comprise both the linear and angular components as shown.

$$\mathrm{v} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \tag{8}$$

$$\dot{\mathrm{v}} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \tag{9}$$

Newton's first law of motion defines inertia as the tendency of the body to remain in its state of rest or of uniform motion unless an external force acts on it to change its state. The inertial component for a rigid body with 6 DoF was represented as shown below.

$$M_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & I_{xy} & I_{xz} \\ 0 & 0 & 0 & I_{xy} & I_{yy} & I_{yz} \\ 0 & 0 & 0 & I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \tag{10}$$

Coriolis effect was the effect experience by a moving body in a rotating system and it acts perpendicular to the direction of motion and towards the center of rotation. The Coriolis component for an underwater vehicle was given below.

$$C_{RB}(v) = \begin{bmatrix} mS(v_2) & -mS(v_2)S(r_g^b) \\ mS(r_g^b)S(v_2) & -S(I_b v_2) \end{bmatrix} \tag{11}$$

Here S represents the skew matrix, v_2 represents the vessel velocity vector and I_b represents the moment of inertia matrix. Since the center of mass coincides with the center of gravity, the skew matrix ($S(r_g^b)$ for the CG vector becomes null. This yield, $C_{RB}(v)v$ as shown below [9].

$$C_{RB}(v)\mathrm{v} = -\begin{bmatrix} m(qw - rv) \\ m(ru - pw) \\ m(pv - qu) \\ qr(I_{yy} - I_{zz}) \\ rp(I_{zz} - I_{zz}) \\ qp(I_{xx} - I_{yy}) \end{bmatrix} \tag{12}$$

2.2.2 HYDRODYNAMIC FORCES

The hydrodynamic forces include three components: the hydrodynamic drag ($\mathrm{D}(v)\mathrm{v}$), Coriolis-centripetal force ($C_A(v)\mathrm{v}$) and inertial force ($M_A\dot{v}$). This was represented as,

$$\tau_{hydrodynamics} = -\mathrm{D}(v)\mathrm{v} - C_A(v)\mathrm{v} - M_A\dot{v} \tag{13}$$

The hydrodynamic drag was the damping force acting on the body and was comprised of linear and quadratic drag as shown.

$$\mathrm{D}(v) = D_{linear} + D_{quadratic} \tag{14}$$

Where D_{linear} and $D_{quadratic}$ were diagonal matrices comprising of the linear and quadratic damping components respectively. Adding the two matrices and multiplying them with the velocity vector gives the damping force as shown.

$$\mathrm{D}(v) = -\begin{bmatrix} (X_u + X_{u|u|}|u|)u \\ (Y_v + Y_{v|v|}|v|)v \\ (Z_w + Z_{w|w|}|w|)w \\ (K_p + K_{p|p|}|p|)p \\ (M_q + M_{q|q|}|q|)q \\ (N_r + N_{r|r|}|r|)r \end{bmatrix} \tag{15}$$

The Coriolis-centripetal force ($C_A(v)\mathrm{v}$) and inertial force ($M_A\dot{v}$) were obtained from Fossen [15] using the low speed assumption as shown below.

$$C_A(v)\mathrm{v} = -\begin{bmatrix} Y_{\dot{v}}vr - Z_{\dot{w}}wq \\ Z_{\dot{w}}wp - X_{\dot{u}}ur \\ X_{\dot{u}}uq - Y_{\dot{v}}vp \\ (Y_{\dot{v}} - Z_{\dot{w}})vw + (M_{\dot{q}} - N_{\dot{r}})qr \\ (Z_{\dot{w}} - X_{\dot{u}})uw + (N_{\dot{r}} - K_{\dot{p}})pr \\ (X_{\dot{u}} - Y_{\dot{v}})uv + (K_{\dot{p}} - M_{\dot{q}})pq \end{bmatrix} \tag{16}$$

$$\dot{M_A}\dot{v} = \begin{bmatrix} X_{\dot{u}}\dot{u} \\ Y_{\dot{v}}\dot{v} \\ Z_{\dot{w}}\dot{w} \\ K_{\dot{p}}\dot{p} \\ M_{\dot{q}}\dot{q} \\ N_{\dot{r}}\dot{r} \end{bmatrix} \tag{17}$$

2.2.3 HYDROSTATIC FORCES

The hydrostatic forces consisted of two forces: buoyancy and gravitational force(weight). It was assumed that the Center of Buoyancy (CB) and Center of Gravity (CG) were on the z-axis and z_B was the distance of the CB to the CG for simplicity as the assumption holds for most of the cases. Buoyancy was the upward force caused when immersing a body into a liquid and was a factor of the volume of the liquid displaced by the body. In the case of a fully submerged vehicle (AUV), the volume of liquid displaced was equal to the total volume of the vehicle. Hence buoyancy (B) and the weight (W) of the vehicle, are as shown.

$$\mathrm{B} = \rho \mathrm{g} \mathrm{V} \tag{18}$$

$$\mathrm{W} = \mathrm{mg} \tag{19}$$

Here, ρ was the density of water and V was the volume of the vehicle. The sum of the two forces gives the hydrostatic force ($\mathrm{g}(\eta)$) which in vector form was as shown.

$$\mathrm{g}(\eta) = -\begin{bmatrix} -(W-B)\sin\theta \\ (W-B)\cos\theta\sin\phi \\ (W-B)\cos\theta\cos\phi \\ z_B\cos\theta\sin\phi \\ z_B\sin\theta \\ 0 \end{bmatrix} \tag{20}$$

2.2.4 THRUSTER FORCES

The final component of the dynamic model involves the thruster forces and other disturbances. Deep underwater, the effects of wind and ocean currents were minimal and hence τ_{wind} and τ_{wave} were equivalent to zero. The thrust forces were the vector sum of the individual thruster forces in the respective axes and given by,

$$T_i = \begin{bmatrix} T_x \\ T_y \\ T_z \\ T_\phi \\ T_\theta \\ T_\psi \end{bmatrix} \quad (21)$$

2.3 DYNAMIC MODEL DERIVATION

Substituting the individual components into the equation 6 and decomposing them to their corresponding acceleration components, we have the dynamic model as follows:

$$\dot{u} = \left(\frac{1}{m - X_{\dot{u}}}\right)\Big(\big(X_u + X_{u|u|}|u|\big)\mathrm{u} + (B - W)\sin\theta + \mathrm{m}(rv - qw) - Y_{\dot{v}}\mathrm{rv} + Z_{\dot{w}}\mathrm{qw} + T_x\Big) \quad (22)$$

$$\dot{v} = \left(\frac{1}{m - Y_{\dot{v}}}\right)\Big(\big(Y_v + Y_{v|v|}|v|\big)\mathrm{v} - (B - W)\sin\phi\cos\theta + \mathrm{m}(pw - ru) - X_{\dot{u}}\mathrm{ru} + Z_{\dot{w}}\mathrm{pw} + T_y\Big) \quad (23)$$

$$\dot{\mathrm{w}} = \left(\frac{1}{\mathrm{m} - \mathrm{Z}_{\dot{\mathrm{w}}}}\right)\Big(\big(\mathrm{Z_w} + \mathrm{Z_{w|w|}}|\mathrm{w}|\big)\mathrm{w} - (\mathrm{B} - \mathrm{W})\cos\theta\cos\phi + \mathrm{m}(\mathrm{qu} - \mathrm{pv}) - \mathrm{X}_{\dot{\mathrm{u}}}\mathrm{qu} + \mathrm{Y}_{\dot{\mathrm{v}}}\mathrm{pv} + \mathrm{T_z}\Big) \quad (24)$$

$$\dot{p} = \left(\frac{1}{I_x - K_{\dot{p}}}\right)\Big(\big(K_p + K_{p|p|}|p|\big)\mathrm{p} - \big(M_{\dot{q}} - N_{\dot{r}}\big)\mathrm{qr} + \big(I_{yy} - I_{zz}\big)\mathrm{qr} + (Z_{\dot{w}} - Y_{\dot{v}})\mathrm{vw} + \mathrm{B}z_B\cos\theta\sin\phi + T_\theta\Big) \quad (25)$$

$$\dot{q} = \left(\frac{1}{I_y - M_{\dot{q}}}\right)\Big(\big(M_q + M_{q|q|}|q|\big)\mathrm{q} - \big(N_{\dot{r}} - K_{\dot{p}}\big)\mathrm{pr} + \big(I_{zz} - I_{yy}\big)\mathrm{pr} + (X_{\dot{u}} - Z_{\dot{w}})\mathrm{uw} + \mathrm{B}z_B\sin\theta + T_\phi\Big) \quad (26)$$

$$\dot{r} = \left(\frac{1}{I_z - N_{\dot{r}}}\right)\Big(\big(N_r + N_{r|r|}|r|\big)\mathrm{r} - \big(K_{\dot{p}} - M_{\dot{q}}\big)\mathrm{pq} + \big(I_{xx} - I_{yy}\big)\mathrm{pq} + (Y_{\dot{v}} - X_{\dot{u}})\mathrm{uv} + T_{\Psi}\Big) \quad (27)$$

These six dynamic equations were generic to an underwater vehicle with 6 DoF. For our purpose, we modified them based on the configuration of the RexROV before using them in the Extended Kalman Filter.

2.4 REDUCED DYNAMIC MODEL FOR RexROV

A Creo based CAD model, as shown in figure 4, was developed based on the information from the UUVSim [2] website.

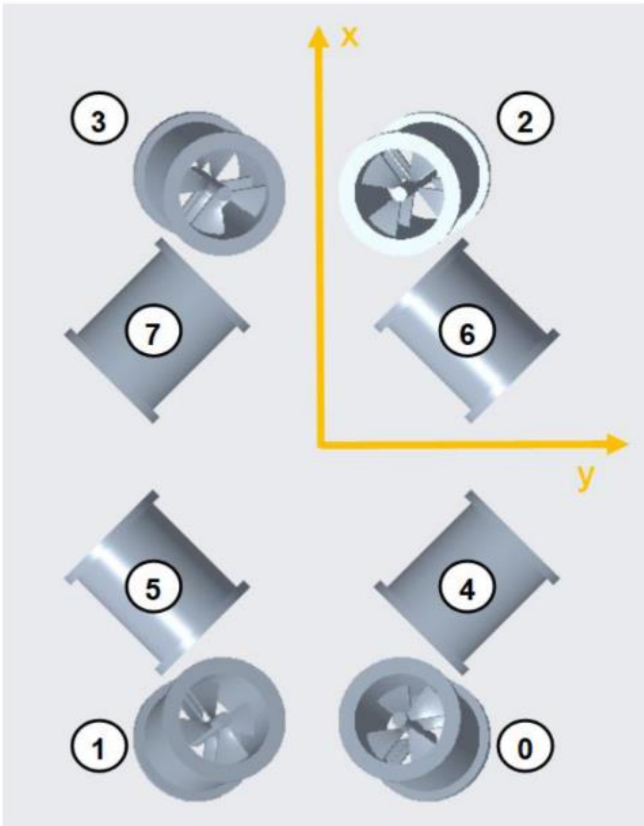

Figure 4: RexROV Thruster Configuration

Table 2 shows the Thruster IDs and their respective location from the CG including their orientations.

Table 2: RexROV Thruster Positions and Orientations

Thruster ID	**Location w.r.t. Body Center (m)**			**Orientation (deg)**		
	lx	**ly**	**lz**	**φ**	**θ**	**ψ**
0	-0.890895	0.334385	-0.528822	0	74.53	-53.21
1	-0.890895	-0.334385	-0.528822	0	74.53	53.21
2	0.890895	0.334385	-0.528822	0	105.47	53.21
3	0.890895	-0.334385	-0.528822	0	105.47	-53.21
4	-0.412125	0.505415	-0.129	0	0	45
5	-0.412125	-0.505415	-0.129	0	0	-45
6	0.412125	0.505415	-0.129	0	0	135
7	0.412125	-0.505415	-0.129	0	0	-135

Through the inherent configuration of the RexROV, the submarine was stable with no pitch and roll due to its structure. From an operations perspective, there was no need for the RexROV or similar vehicles to possess pitch and roll capabilities. Thus 2 DoF can be eliminated and the 6 DoF model can be reduced to a pseudo 4 DoF model similar to works performed in [16] and [17]. This means that φ and θ were always 0. Likewise, their corresponding velocities, p and q, and accelerations, $\dot{p}$ and $\dot{q}$, components were always 0. The individual thruster forces T_0 to T_7 were resolved into their respective components along x, y and z axis based on their orientations and locations from the center of mass to calculate the thrust forces needed for the dynamic model. The thrust forces of RexROV was given by the following equations.

$$T_x = \sum_{i=0}^{3} T_i \cos \theta_i \cos \psi_i + \sum_{j=4}^{7} T_j \cos \psi_j \tag{28}$$

$$T_y = \sum_{i=0}^{3} T_i \cos \theta_i \cos \psi_i + \sum_{j=4}^{7} T_j \sin \psi_j \tag{29}$$

$$T_z = \sum_{i=1}^{3} T_i \sin \psi_i \tag{30}$$

$$T_\psi = \sum_{i=0}^{7} T_i (lx_i \sin \psi_i - ly_i \cos \psi_i) \tag{31}$$

The dynamic model for RexROV reduced to the following set of equations (32) to (35).

$$\dot{u} = \left(\frac{1}{m - X_{\dot{u}}}\right)\left(\left(X_u + X_{u|u|}|u|\right)\mathrm{u} + \mathrm{mrv} - Y_{\dot{v}}\mathrm{rv} + T_x\right) \tag{32}$$

$$\dot{v} = \left(\frac{1}{m - Y_{\dot{v}}}\right)\left(\left(Y_v + Y_{v|v|}|v|\right)\mathrm{v} + \mathrm{mru} + X_{\dot{u}}\mathrm{ru} + T_y\right) \tag{33}$$

$$\dot{w} = \left(\frac{1}{m -_Z \dot{w}}\right)\left(\left(Z_w + Z_{w|w|}|w|\right)w - (B - W) + T_z\right) \tag{34}$$

$$\dot{r} = \left(\frac{1}{I_z - N_{\dot{r}}}\right)\left(\left(N_r + N_{r|r|}|r|\right)\mathrm{r} + (Y_{\dot{v}} - X_{\dot{u}})\mathrm{uv} + T_{\Psi}\right) \tag{35}$$

The physical parameters and the coefficient terms for RexROV in the dynamic equations were given in table 3 and table 4, respectively.

Table 3: RexROV Physical Parameters

Parameter	**Value**
Length(x-axis)	2.6 m
Width(y-axis)	1.5 m
Height(z-axis)	1.6 m
Mass(m)	1.863 kg
Volume(V)	1.838 m3
B	18393.9972 N
g	9.81 m/s
r	1000 kg/m3
Izz	691.23 kg m2

Table 4: RexROV Coefficients

Coefficient	Value
$X_{\dot{u}}$	779.79
$Y_{\dot{v}}$	1222
$Z_{\dot{w}}$	3959.9
$K_{\dot{p}}$	534.9
$M_{\dot{q}}$	842.69
$N_{\dot{r}}$	224.32
X_u	-74.82
Y_v	-69.48
Z_w	-782.4
K_p	-268.8
M_q	-309.77
N_r	-105
$X_{u\|u\|}$	-748.22
$Y_{v\|v\|}$	-992.53
$Z_{w\|w\|}$	-1821.01
$K_{p\|p\|}$	-672
$M_{q\|q\|}$	-774.44
$N_{r\|r\|}$	-523.27

2.5 IMU ACCELERATION TEST

As presented in equations (32) to (35), the dynamic model for RexROV involves calculating the acceleration values. The accuracy of the modeling can be verified by comparing it to the acceleration values generated by the IMU sensor on UUVSim. The IMU sensor accommodates drifting bias, following a random walk with zero-mean white Gaussian noise as the rate. Its model includes zero-mean Gaussian noise as well. The following figures 5 and 6, shows the acceleration values generated by the model and the acceleration values given by the IMU sensor in the simulation in the x-axis and y-axis for behavior evaluation 3 test course [11].

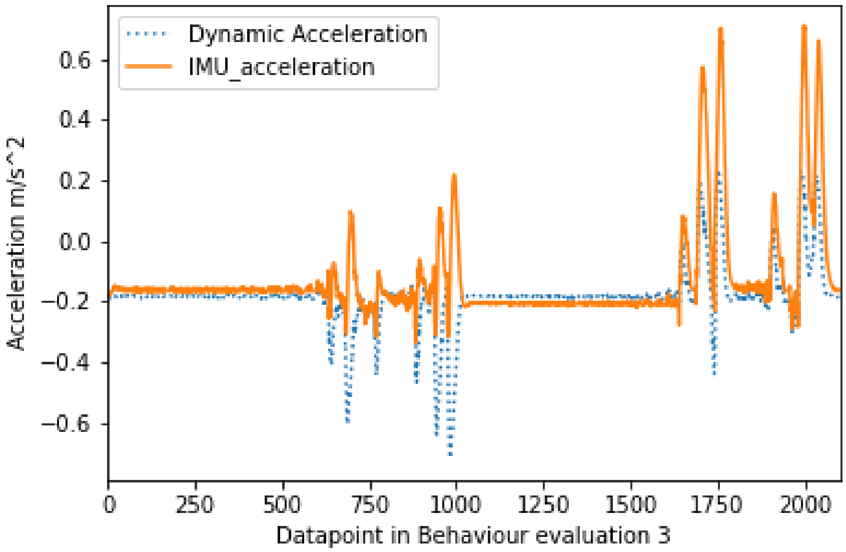

Figure 5 : Acceleration in x-axis

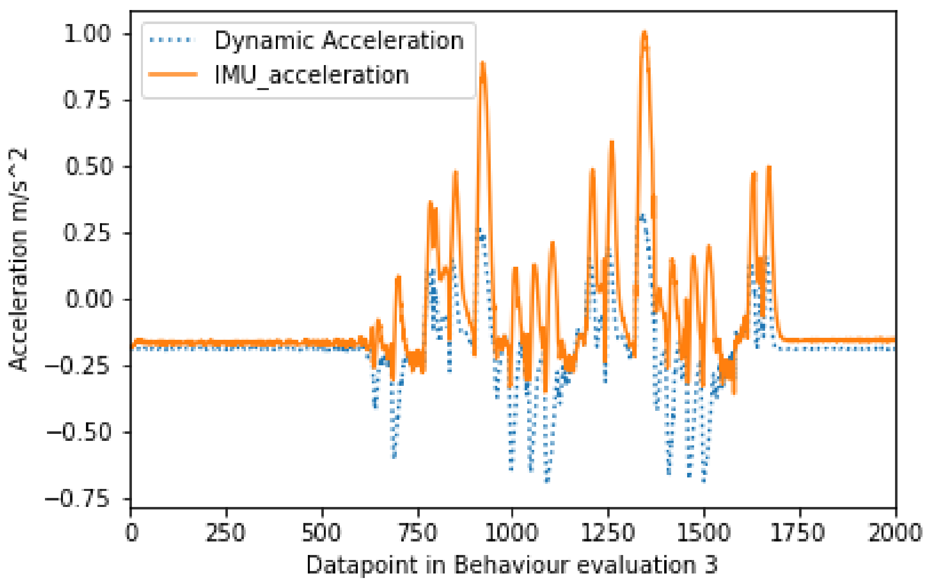

Figure 6 : Acceleration in y-axis

Our model performs similarly to the IMU sensor. It was also important to note that the IMU sensor modeled in the simulation had a very high level of accuracy with a variance of 0:004m/s^2. However, in practice, the IMU sensors have a higher variance than the one in simulation. In general, IMUs were prone to drift, which results in error accumulation over time.

This results in divergence with actual values in practice while calculating position. As seen in figure 3 and 4, the dynamic model provided a change in the value where there was a change in the trajectory. The positive acceleration values calculated by the model were consistent with the IMU sensor values. Further research will be conducted to understand the inconsistency on the negative acceleration values calculated by the model. The dynamic model has an edge over the IMU sensor in the fact that it uses velocity and thrust forces for calculation. The thrust forces can be reliably received from the generated look-up table based on the thruster RPM. The velocity, being the first derivative, gives a reliable estimate for the position calculation as opposed to acceleration, which is the second derivative.

CHAPTER 3: COMPARISION OF DYNAMIC AND KINEMATIC KALMAN FILTER

3.1 KALMAN FILTERS

Kalman filter-based localization techniques have been prevalent since the 1960s [4]. The Kalman Filter was an iterative mathematical tool used to estimate the position of a system based on a system of equations and measurements. The Kalman filter has three steps namely Initialization, Prediction and Update step.

Initialization: Initialize the state of the filter and the belief in the state

Prediction:

1. Use the process model to predict the state of the system in the next time step
2. Adjust belief to account for uncertainty in prediction

Update:

1. Get a measurement from the sensor and associated belief about its accuracy
2. Compute residual between predicted state and measurement
3. Compute Kalman Gain based on whether the measurement or prediction was more accurate
4. Set state between the prediction and measurement based on scaling factor
5. Update belief in the state based on how certain we were in the measurement

Initialization was a one-time process while the prediction and update were iterative. The Kalman filter has two major conditions in order to work:

1. The Kalman Filter works with Gaussian Distribution
2. The prediction and the update equations were linear

These two conditions go hand in hand. A linear system fed to a gaussian produces a gaussian output while a nonlinear system produces a non-gaussian output. Hence Kalman Filters work well with linear systems of equations. However, most of the real-time systems that we encounter were often nonlinear. One way to apply a Kalman Filter to these non-linear systems was to approximate them to linear systems using Taylor series around the mean of the nonlinear Gaussian. This version of the Kalman filter is referred to as the Extended Kalman filter (EKF). While the equations remain the same from the Kalman Filter, one major change was that we used the Gauss Jacobian Matrix instead of the State transition matrix for linear approximations. At each time step, the Jacobian was evaluated with current predicted states. These matrices were used in the Kalman filter equations. This process linearizes the nonlinear function around the current estimate. The EKF algorithm mentioned below was a first order EKF. Higher order EKF can be obtained by retaining higher order terms in the Taylor series. The EKF algorithm was as shown below.

Algorithm 1 Extended Kalman Filter

Result: State Matrix x

Initialization: Initialize the state of the filter (P, Q) and the belief in the state (K)

While New Datapoint **do**

Predict:

$\widehat{x_k} = f(x, u)$

$F_k = \frac{\partial f(x,u)}{\partial x}|_{x,u}$

$\widehat{P_k} = F_k P_{k-1} F_k^T + Q$

Update:

$y_k = z_k - h(\widehat{x_k})$

$H_k = \frac{\partial h(\hat{x})}{\partial x}|_x$

$S_k = H_k \widehat{P_k} H_k^T + R$

$K_k = \widehat{P_k} H_k^T S_k^{-1}$

$x_k = \widehat{x_k} + K_k y_k$

$P_k = (I - K_k H_k)\widehat{P_k}$

end

The terms used in algorithm 1 are listed below.

x - State Matrix

$\hat{x}$ - State Prediction

$f\,(x,\, u)$ - Function to predict the states

F - State Transition Matrix

$\hat{P}$ - Predicted Covariance

P - Process Noise Covariance

Q - Measurement Noise Covariance

z - Measurement from Sensors

$h(\hat{x})$ - Prediction to Measurement coordinate

y - Difference between Prediction and Measurement

H - Jacobian Matrix

R - Measurement Noise Covariance

K - Kalman Gain

I - Identity Matrix

k - Represents the current time step

k-1 - Represents the previous time step

Another version of Kalman filter that exists was the Unscented Kalman Filter (UKF. A UKF differs from an EKF in that it takes a defined set of multiple points from the nonlinear system, including the mean, for approximation while an EKF considers only the mean to approximate the system [5]. Thus, UKF offers a better approximation of the non-linear system than the EKF. The defined set of multiple points were called the sigma points and each of these points have a weight associated with them. The sigma points were representative of the whole distribution. The number of sigma points taken per dimension was given by (2N+1), where N was the number of dimensions of the system. After selecting the sigma points, the weights associated with each of the points were calculated. These points were then transformed using a nonlinear function. The gaussian of the resulting transformed points were calculated and its mean and variance are computed. This method is known as the Unscented method. To summarize, the Unscented method consists of the following five steps:

1. Compute Set of Sigma Points
2. Assign Weights to each sigma point
3. Transform the points through nonlinear function
4. Compute Gaussian from weighted and transformed points
5. Compute Mean and Variance of the new Gaussian.

While taking all the points from the nonlinear system would give a better approximation compared to taking a defined set of points, the method fails in terms of computation power and optimality. So, for our purposes, the Extended Kalman filter was used. The EKF uses a process model to predict the system states.

3.2 EXTENDED KALMAN FILTER FOR RexROV

Applying the EKF algorithm 1 to the RexROV submarine, we have our state matrix for 4 DoF as shown below.

$$x = \begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix} \tag{36}$$

The state transition matrix F was as follows.

$$F = \begin{bmatrix} \frac{\partial f(x)}{\partial x} & \frac{\partial f(x)}{\partial y} & \frac{\partial f(x)}{\partial z} & \frac{\partial f(x)}{\partial \psi} \\ \frac{\partial f(y)}{\partial x} & \frac{\partial f(y)}{\partial y} & \frac{\partial f(y)}{\partial z} & \frac{\partial f(y)}{\partial \psi} \\ \frac{\partial f(z)}{\partial x} & \frac{\partial f(z)}{\partial y} & \frac{\partial f(z)}{\partial z} & \frac{\partial f(z)}{\partial \psi} \\ \frac{\partial f(\psi)}{\partial x} & \frac{\partial f(\psi)}{\partial y} & \frac{\partial f(\psi)}{\partial z} & \frac{\partial f(\psi)}{\partial \psi} \end{bmatrix} \tag{37}$$

For simplification purpose, we assume that the other axis elements do not affect the current axis elements. Using the previously discussed stability assumption, the non-diagonal elements of the F-matrix become null as shown,

$$F = \begin{bmatrix} \frac{\partial f(x)}{\partial x} & 0 & 0 & 0 \\ 0 & \frac{\partial f(y)}{\partial y} & 0 & 0 \\ 0 & 0 & \frac{\partial f(z)}{\partial z} & 0 \\ 0 & 0 & 0 & \frac{\partial f(\psi)}{\partial \psi} \end{bmatrix} \tag{38}$$

The state transition matrix can be obtained using Newton's equations of motion.

$$f(x) = x_{k-1} + v_k \Delta t + \frac{\dot{v_k} \Delta t^2}{2} \tag{39}$$

Here, v_k was the velocity matrix and $\dot{v_k}$ was the acceleration matrix as shown in equations (8) and (9). We derived the first term in the F-matrix and generalize for the other three terms.

$$\frac{\partial f(x)}{\partial x} = 1 + \frac{\partial u}{\partial x} \Delta t + \frac{\partial \dot{u}}{\partial x} \frac{\Delta t^2}{2} \tag{40}$$

$$\frac{\partial f(x)}{\partial x} = 1 + \frac{\partial u}{\partial t}\frac{\partial t}{\partial x}\Delta t + \frac{\partial \dot{u}}{\partial t}\frac{\partial t}{\partial x}\frac{\Delta t^2}{2} \qquad (41)$$

The filter we were considering for the simulation worked at a frequency of 10Hz. For such a small change in time, the partial derivative term was taken as shown below.

$$\frac{\partial f(x)}{\partial x} = 1 + \dot{u}\frac{1}{u}\Delta t + \frac{\Delta \dot{u}}{\Delta t}\frac{1}{u}\frac{\Delta t^2}{2} \qquad (42)$$

This can be further simplified to,

$$\frac{\partial f(x)}{\partial x} = 1 + \frac{\dot{u}}{u}\Delta t + \frac{\Delta \dot{u}\Delta t}{2u} \qquad (43)$$

Similarly, the other three terms for the F-matrix were derived.

$$\frac{\partial f(y)}{\partial y} = 1 + \frac{\dot{v}}{v}\Delta t + \frac{\Delta \dot{v}\Delta t}{2v} \qquad (44)$$

$$\frac{\partial f(z)}{\partial z} = 1 + \frac{\dot{w}}{w}\Delta t + \frac{\Delta \dot{w}\Delta t}{2w} \qquad (45)$$

$$\frac{\partial f(\psi)}{\partial \psi} = 1 + \frac{\dot{r}}{r}\Delta t + \frac{\Delta \dot{r}\Delta t}{2r} \qquad (46)$$

In the simulation, the velocity values were obtained from the Doppler Velocity Log (DVL) sensor while the acceleration values were obtained from the Inertial Measurement Unit (IMU) sensor. The Global Positioning System (GPS) sensor provided the x and y measurements. The pressure sensor gave the z measurements and the compass built into the IMU sensor gave the ψ update. It was important to note that both the predictions ($h(\hat{x})$) and the measurements ($\hat{x}$) were in the same coordinates. This rendered the Jacobian matrix (H) as an Identity matrix of size 4.

$$H = \frac{\partial h(\hat{x})}{\partial \hat{x}} = \frac{\partial \hat{x}}{\partial \hat{x}} = I_4 \qquad (47)$$

The update equation was further simplified as shown.

$$S_k = \widehat{P_k} + R \qquad (48)$$

$$K_k = \widehat{P_k} S_k^{-1} \qquad (49)$$

$$x_k = \widehat{x_k} + K_k y_k \tag{50}$$

$$P_k = (I - K_k)\widehat{P_k} \tag{51}$$

The Extended Kalman filter algorithm for RexROV after applying the simplifications is shown below.

Algorithm 1 Extended Kalman Filter for RexROV

Result: State Matrix x

Initialization: Initialize the state of the filter (P, Q) and the belief in the state (K)

While New Datapoint **do**

Predict:

$\widehat{x_k} = f(x,u)$

$F_k = \frac{\partial f(x,u)}{\partial x}|_{x,u}$

$\widehat{P_k} = F_k P_{k-1} F_k^T + Q$

Update:

$y_k = z_k - (\widehat{x_k})$

$H_k = \frac{\partial h(\hat{x})}{\partial x}|_x$

$S_k = \widehat{P_k} + R$

$K_k = \widehat{P_k} S_k^{-1}$

$x_k = \widehat{x_k} + K_k y_k$

$P_k = (I - K_k)\widehat{P_k}$

end

The following algorithm was implemented in real-time in UUVSim and the Kalman filter prediction values were used in a pure pursuit controller implemented by the Submarine Bois team to drive the submarine on a predetermined path [11].

3.3 PURE PURSUIT CONTROLLER

For a given set of waypoints, a pure pursuit algorithm approaches the target waypoint ignoring the dynamics/kinematics of the environment and the vehicle (except vehicle length). Once in the vicinity (user defined) of the target waypoint, the algorithm generates the steering command for the next waypoint until the system reaches the last waypoint [25]. For our purposes, we were generated the steering command while the linear velocity remained constant.

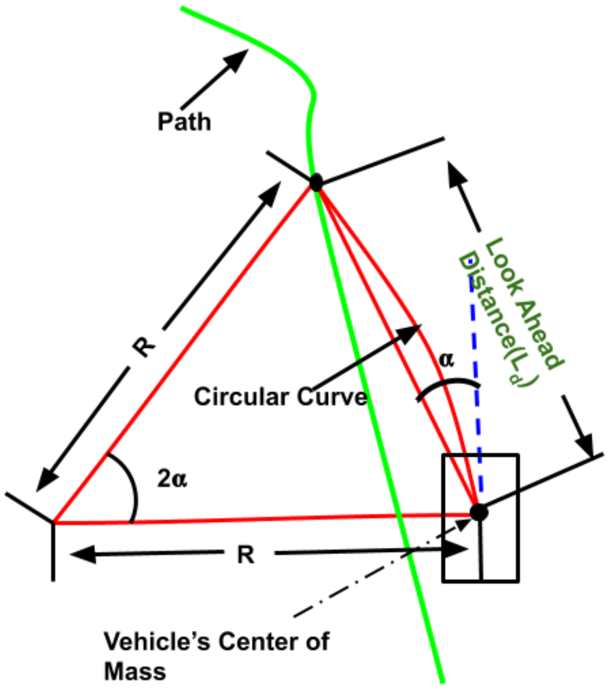

Figure 7: Pure Pursuit Model

Considering RexROV as a point mass [26], Pure Pursuit aims to align the Center of Mass (in our case) along the reference trajectory through generating steering commands using the following formulations.

$$K = \frac{2 * sin(\alpha)}{L_d} \tag{52}$$

$$\delta = tan^{-1}(K * L) \tag{53}$$

The vehicle length (L) and the look ahead distance (L_d) were taken as unity as well. α was the difference between vehicle pose and the slope of the line connecting the target point and the last waypoint. x_k in the EKF algorithm 2 gives the current pose of the RexROV. Both slope and pose were converted within a 0 to 2p range. Output δ was published to the cmdvel Rostopic with a gain of 0.3.

3.4 TEST COURSES

In this section, the performance of the kinematic and the dynamic model driven pure pursuit controllers were compared on various test courses adopted from [11]. The figure shows the actual path followed by the vehicle as opposed to the output of the dynamic model and kinematic model. The results from the test courses were summarized as a table. Total was the mean of the Cartesian distance between the reference path and the path taken by the RexROV when using the controllers. X-Kalman gives the mean squared error (MSE) between the value predicted by the Kalman filter and the actual path traced by the RexROV. Likewise, Y-Kalman gives MSE on the y-axis.

3.4.1 BEHAVIOR EVALUATION 1

The first test course involved moving along a straight line followed by a fixed turning radius and a sharp turn.

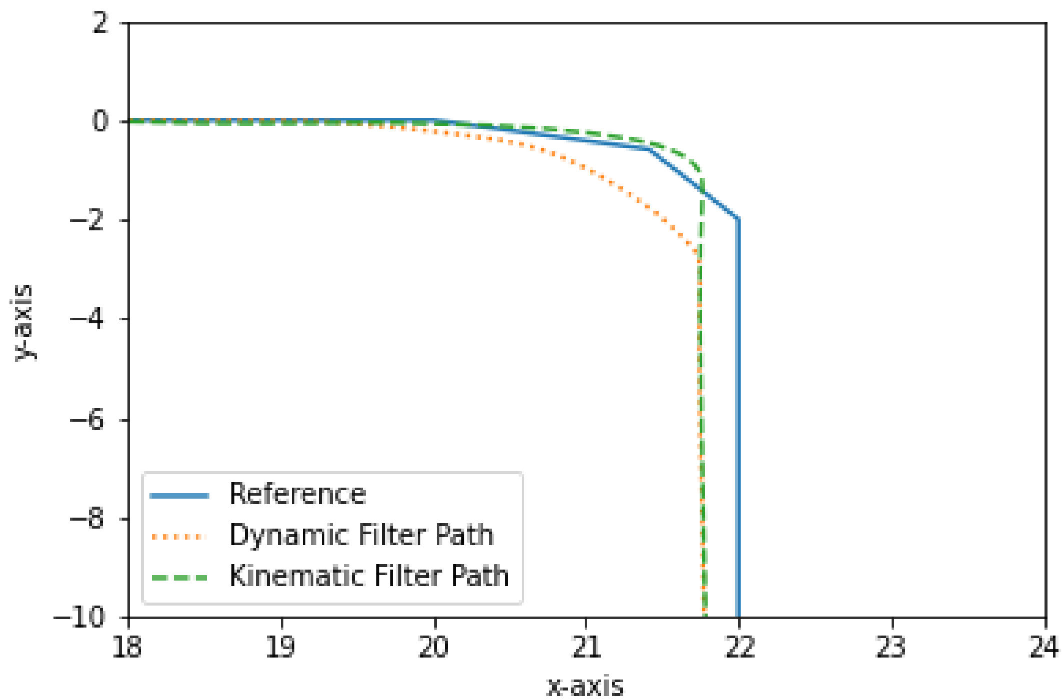

Figure 8: Behavior Evaluation 1

As seen in figure 6, the kinematic filter followed the reference path much closer than the dynamic filter. Initially, there was no divergence from the path. After the simple turn, both the filters converge at the same rate.

Table 5: Results of Behavior Evaluation 1

Error	Dynamic Filter	Kinematic Filter
Total	0.1166m	0.09880m
X-Kalman	0.0825m	0.3432m
Y-Kalman	0.2665m	1.024m

From table 5, the total error of the kinematic filter was less than the dynamic filter. However, this was attributed to the controller and not the filter itself as it can be inferred from the other two entries in the table. To make further assessment of the filters, they were tested on the behavior evaluation 2 test course.

3.4.2 BEHAVIOR EVALUATION 2

The second test course was a sinusoidal path which used curves of constant radius with both clockwise and counterclockwise directions. This test course helps in determining if there were handling and navigation issue.

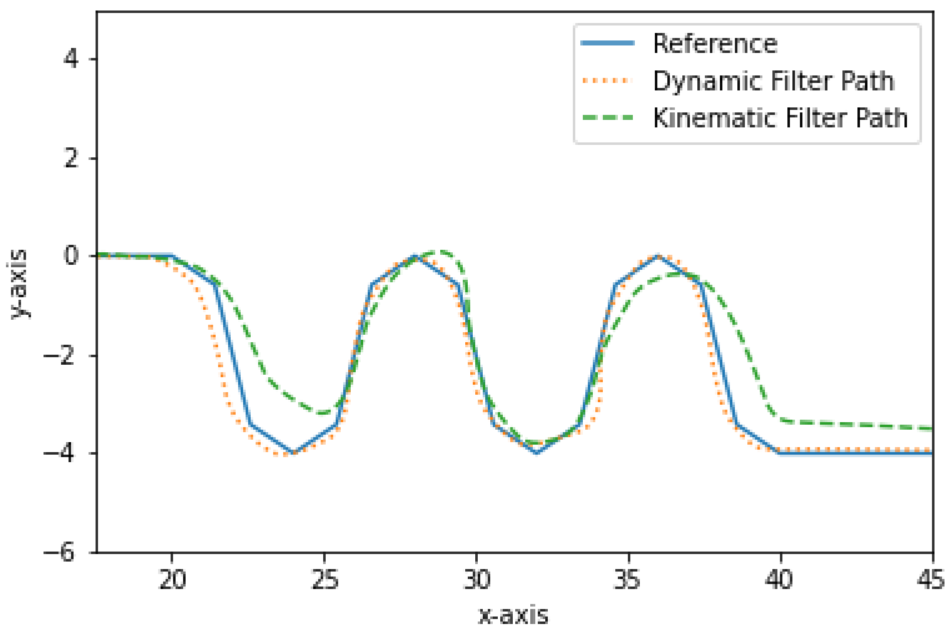

Figure 9: Behavior Evaluation 2

As seen in figure 7, the dynamic filter does a much better job than the kinematic filter when tracing a curve. The dynamic filter converged faster when there was a change in both x and y co-ordinates.

Table 6 : Results of Behavior Evaluation 2

Error	Dynamic Filter	Kinematic Filter
Total	0.1659m	0.2482m
X-Kalman	0.1213m	0.39145m
Y-Kalman	0.3682m	4.336m

From table 6, the mean error of the dynamic filter was less than the kinematic filter. The dynamic filter handled the simultaneous changes in the x-axis and y-axis much better than the kinematic filter.

3.4.3 BEHAVIOR EVALUATION 3

The third test course was a U-shaped path with clockwise and counterclockwise curves and straightaways. These curves were of equal radii and the straightaways were horizontal and vertical. Only at the curves, both the x-axis and y-axis values change.

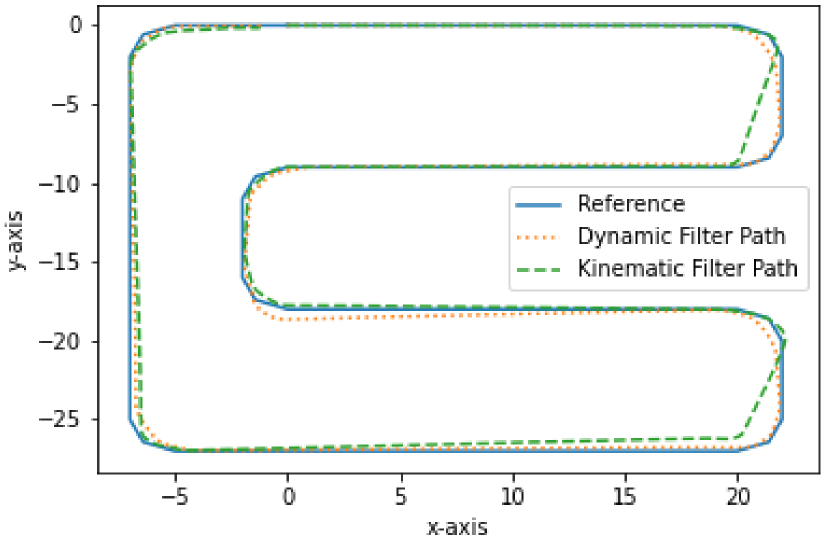

Figure 10: Behavior Evaluation 3

Table 7 : Results of Behavior Evaluation 3

Error	Dynamic Filter	Kinematic Filter
Total	0.0738m	0.1965m
X-Kalman	0.2462	0.4028
Y-Kalman	0.4003m	4.0014m

From figure 8 and table 7, in behavior evaluation course 3, which includes all possible movements of the ROV, the dynamic filter converges to the path more quickly than its kinematic counterpart.

CHAPTER 4: NEURAL NETWORK DYNAMIC MODEL

4.1 MULTI-LAYER NEURAL NETWORKS

Neural networks have been used to make classifications, predictions and a variety of other applications [18][19]. Neural networks have an input later and an output layer. The layers in between the input and output layers were referred to as hidden layers. Depending on the number of hidden layers used, they can be classified as shallow and deep neural networks. Like any machine learning algorithm, the neural network takes in a set of inputs and a set of target values to make predictions. Graphically, a multi-layer neural network can be represented as shown.

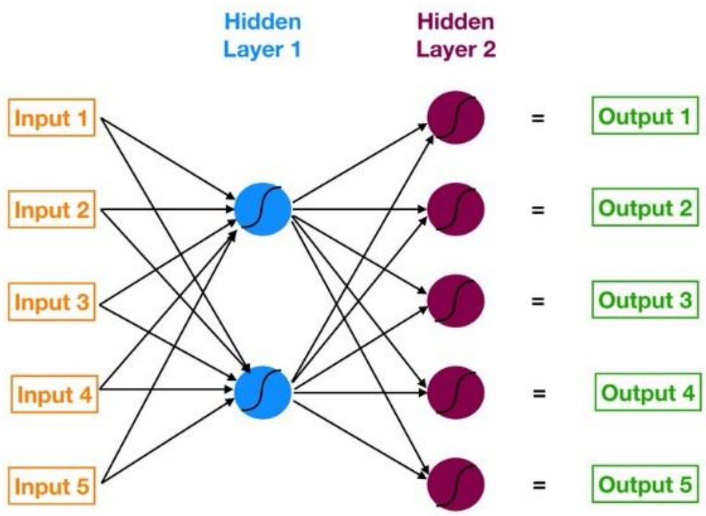

Figure 11: Multi-layer neural network [20]

Mathematically, a multi-layer neural network can be expressed as follows.

$$Z^1 = W^1x + b^1 \tag{48}$$

$$H^1 = \sigma(Z^1) \tag{49}$$

$$Z^l = W^lH^{l-1} + b^l \text{ , l = 2, ……L} \tag{50}$$

$$H^l = \sigma(Z^l) \tag{51}$$

$$U = W^{L+1}H^L + b^{L+1} \tag{52}$$

The neural network takes an input x of size d and produces an output of size K. The matrix Z $\epsilon\ \mathbb{R}^{\mathrm{d_H}}$ represents a linear transformation. W $\epsilon\ \mathbb{R}^{\mathrm{d_H x d}}$ represents the weight matrix and b $\epsilon\ \mathbb{R}^{\mathrm{d_H}}$

was the bias matrix. d_H was the dimension of the hidden layer. To learn the model, an element-wise nonlinearity $\sigma(-) : \mathbb{R} \rightarrow \mathbb{R}$ was applied to each element of matrix Z to obtain the hidden layer H ϵ $\mathbb{R}^{d_H}$. L was the number of hidden layers. $U \epsilon \mathbb{R}^{K}$ represents the output layer.

4.2 ACTIVATION FUNCTIONS

The common choices for the nonlinear activation functions, $\sigma(-)$ were:

Sigmoid - $\sigma(x) = \dfrac{e^z}{1 + e^z}$

Tanh() - $\sigma(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$

Rectified Linear Units (ReLU) - $\sigma(x) = max(x, 0)$

Sigmoid function was one of the most used activation functions and is applicable to the universal approximation theorem [21]. A sigmoid function graph is shown in figure 10.

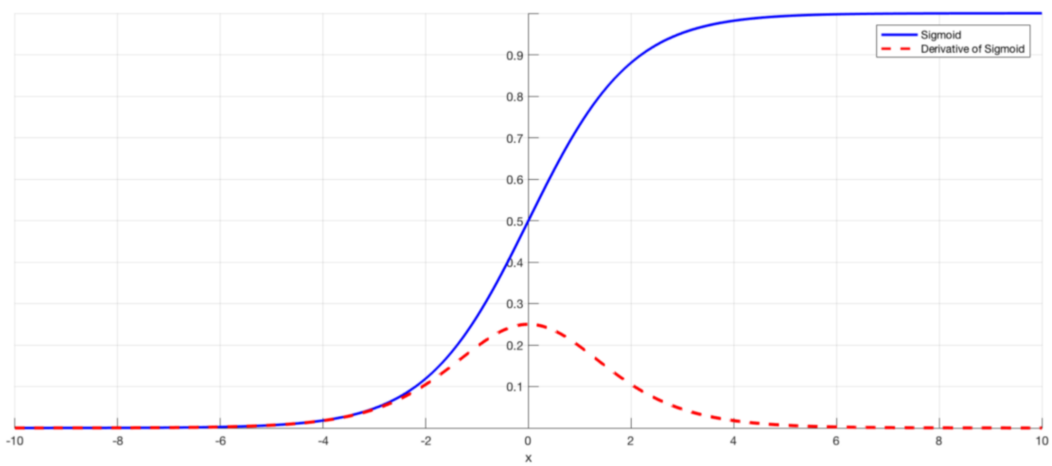

Figure 12: Sigmoid Function and Derivative

The sigmoid function forces the value of x to stay between 0 and 1. Between -2 to +2 values of x, the sigmoid function value changes rapidly and hence there was a bigger gradient in this region. Towards the end of the function, the sigmoid reacts very slowly to changes in x and the gradient is very minimal. In other words, towards the end of the function the gradient becomes negligible and this was called the vanishing gradient problem.

The hyperbolic tangent function or tanh() function graph was shown below.

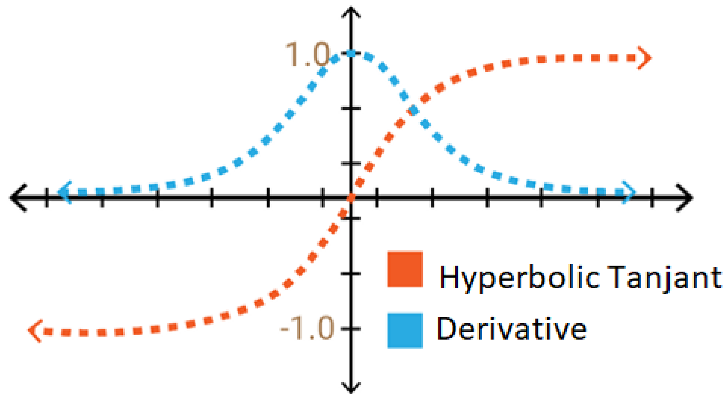

Figure 13: Tanh() function and its derivative

The tanh() function pushes the value of x between -1 and +1, thus giving a wider range of operation compared to the sigmoid function. But the tanh() function also suffers from a similar vanishing gradient problem as the sigmoid function, though not as pronounced.

Rectified Linear Units or ReLU activation functions address the vanishing gradient problem to some extent. In deep neural networks, with lots of neurons, the sigmoid and tanh functions activate every neuron in the same way and their activation was intense. In the case of ReLU, some of the neuron values become zero and hence the computational speed was increased considerably. Although the ReLU improves the computational speed, it suffers from a drawback in that learning does not happen in the regions where the neuron values become zero. The ReLU function was as shown.

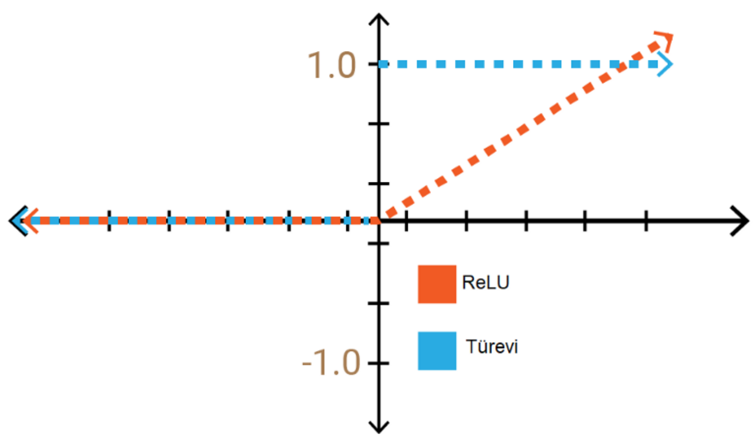

Figure 14: ReLU function and its derivative

There were other activation functions such as leaky ReLU, Exponential Linear Units (ELU), Scaled ELU and Swish functions. The effect of these nonlinear functions on our dynamic model will be discussed.

4.3 GRADIENT DESCENT

Neural networks estimate a statistical model for the relationship between an input X and an output Y. Assume there is dataset (X, Y) $\epsilon\ \mathbb{R}^d$ x Y and a statistical model, f(x,θ) : $\mathbb{R}^d \rightarrow \mathbb{R}^K$. θ ϵ Θ are the parameters in the model and must be estimated. We wish to find a model f(x,θ) such that f(X,θ) is an accurate prediction for Y. We define our objective function as,

$$\mathcal{L}(\theta) = E_{X,Y}[\rho(f(X;\theta), Y)] \tag{53}$$

θ is the set of parameters which minimizes the objective function.

$$\theta = \arg \min_{\theta' \, \epsilon \, \Theta} \mathcal{L}(\theta') \tag{54}$$

The objective function can be minimized using gradient descent, the most popular choice for training neural networks to obtain parameters θ. The gradient descent method is as shown below.

$$\theta^{(l+1)} = \theta^{(l)} - \alpha^{(l)} \, \nabla_\theta \left(\mathcal{L}\left(\theta^{(l)}\right)\right) \tag{55}$$

Here, ∇ is the gradient of the objective function in the steepest direction. The following gradient descent algorithms were popularly used.

- Stochastic gradient descent [21]
- RMSprop [23]
- ADAM [24]
- AdaGrad [22]

4.4 BACK PROPAGATION

The gradient descent algorithms require the calculation of the gradients of ρ with respect to the parameter set θ. For a multi-layer neural network mentioned previously, the parameter set was as follows

$$\theta = \{ W^1, \ldots., W^L, b^1, \ldots., b^L \} \tag{56}$$

Let $\rho \coloneqq \rho(f(X;\theta), Y)$. We define,

$$\delta^l \coloneqq \frac{\partial \rho}{\partial H^l} \tag{57}$$

By chain rule, for *l* = 1,….,L-1,

$$\delta_i^l = \sum_{j=1}^{d_H} \delta_j^{l+1} \frac{\partial H_j^{l+1}}{\partial H_i^l} \tag{58}$$

$$= \sum_{j=1}^{d_H} \delta_j^{l+1} \sigma'\left(Z_j^{l+1}\right) W_{j,i}^{l+1} \tag{59}$$

$$= \left(\delta_j^{l+1} \odot \sigma'\left(Z_j^{l+1}\right)\right)^T W_{:,i}^{l+1} \tag{60}$$

In matrix terms, for *l* = 1,….,L-1,

$$\delta^l = (W^{l+1})^T \left(\delta^{l+1} \odot \sigma'(Z^{l+1})\right) \tag{61}$$

Similarly,

$$\delta^L = (W^{L+1})^T \frac{\partial \rho}{\partial U} \tag{62}$$

Following the same pattern, we have for *l* = 1,….,L-1,

$$\frac{\partial \rho}{\partial b^l} = \delta^l \odot \sigma'(Z^l) \tag{63}$$

$$\frac{\partial \rho}{\partial W^l} = (\delta^l \odot \sigma'(Z^l))(H^{l-1})^T \tag{64}$$

$$\frac{\partial \rho}{\partial b^{L+1}} = \frac{\partial \rho}{\partial U} \tag{65}$$

$$\frac{\partial \rho}{\partial W^{L+1}} = \frac{\partial \rho}{\partial U} (H^L)^T \tag{66}$$

These partial derivatives were used in the gradient descent algorithms previously discussed.

4.5 NEURAL NETWORK DYNAMIC MODEL

A multi-layer neural network was used to model the dynamics of the system. The inputs to the neural network were the same as the inputs given to the dynamic model discussed in Chapter 3. The neural network used the velocities of the system and the thrust forces to predict the acceleration values. For training purpose, we could have used either the acceleration values from the IMU or the output of the dynamic model. As the purpose of our neural network was to replace the dynamic model, we used the acceleration values from the IMU. In the prediction process, we had two types. One architecture, referred to as 4-D model, attempted to predict the four-dimensional dynamic values, specifically, the four acceleration values simultaneously. The other architecture involved four 1-D models and each model predicted an acceleration value. After several empirical iterations, the following structure was selected.

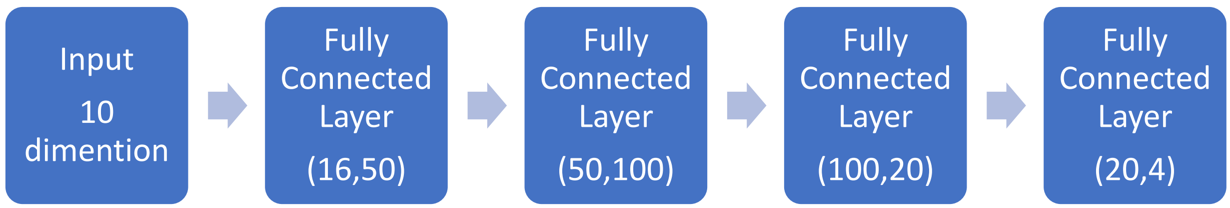

Figure 15: 4 DoF Architecture

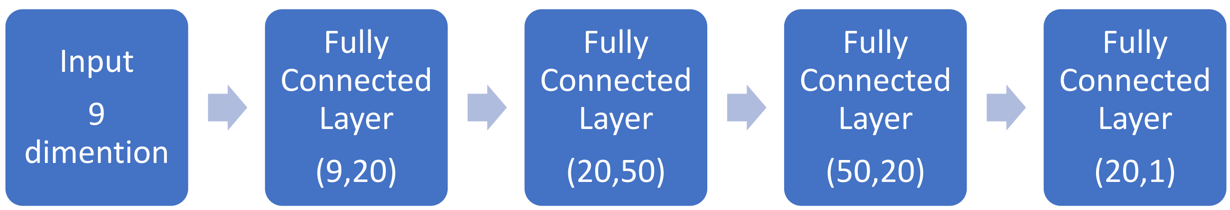

Figure 16: 1 DoF Architecture

After each fully connected network, a nonlinear function was applied and at the last fully connected layer, the predictions were performed. The training dataset generated by making the RexROV follow a random path using randomly generated velocity values. The number of epochs was chosen based on the gradient descent algorithm used, since different algorithms require different convergence times. After training the outputs from the model were used in the EKF algorithm 2 discussed in chapter 3.

4.6 RESULTS ON THE TEST DATASET

The performance of various gradient descent algorithms and nonlinear functions were compared with each other to select the best architecture. A random test dataset obtained by sending varying velocity inputs at the rate of 1 Hz was used to test the model. The outputs of the neural networks were compared to the IMU sensor values and a mean squared error value (m/s^2) was presented in tabular form. The random test course used to evaluate the model is shown in figure 17.

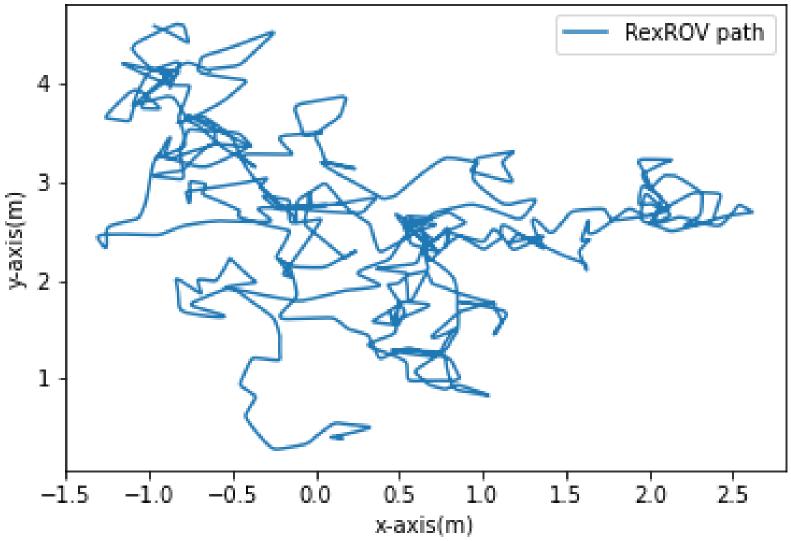

Figure 17: Test Course path

The Mean squared error values for the one-dimensional model on different architecture was presented below.

Table 8: MSE of 1-D model in m/s^2

Non-Linearity	Stochastic Gradient Descent	ADAM	RMS Prop	AdaGrad
Sigmoid	0.392789	0.358554	0.641691	0.590308
Tanh	0.474864	0.443763	0.595621	0.564026
ReLU	0.595786	1.374891	0.520794	0.609667
Leaky ReLU	1.336792	1.707493	0.748638	1.05258
ELU	1.847483	16.76965	0.579572	0.428673
Scaled ELU	0.528643	0.517554	0.505096	**0.214801**
Swish	1.504296	1.134492	1.749343	0.945114

The Mean squared error values for the four-dimensional model on different architecture is represented in table 9. The MSE was obtained as an average of the output from the two models.

Table 9: MSE of 4-D model output in m/s^2

Non-Linearity	Stochastic Gradient Descent	ADAM	RMS Prop	AdaGrad
Sigmoid	0.344719	0.385645	0.594746	0.590196
Tanh	0.467479	0.398121	0.640974	0.589978
ReLU	0.520794	1.607941	0.517014	0.344869
Leaky ReLU	1.739374	5.101167	0.509121	0.494125
ELU	1.67696	1.118611	0.480766	0.238511
Scaled ELU	0.564937	1.209875	0.283453	**0.211873**
Swish	1.515001	2.00879	0.548072	0.50399

4.7 ARCHITECTURE SELECTION

From table 8, the MSE of the 1-D model for AdaGrad algorithm and Scaled ELU nonlinear functions were the smallest value on the test dataset. Similarly, from table 9, the MSE of the 4-D model for AdaGrad algorithm and Scaled ELU nonlinear functions were the smallest on the test dataset. A deeper analysis of the two models for the three test courses mentioned in chapter 3 is presented.

For the behavior evaluation 1 test course, the results are presented in table 10.

Table 10: 1-D model vs 4-D model for BE1

	$\dot{u}$	$\dot{v}$	$\dot{w}$	$\dot{r}$
1-D Model	0.046861	0.022624	0.047363	0.042979
4-D Model	0.065858	0.055501	0.092417	0.093234

For the behavior evaluation 2 test course, the results are presented in table 11.

Table 11: 1-D model vs 4-D model for BE2

	$\dot{u}$	$\dot{v}$	$\dot{w}$	$\dot{r}$
1-D Model	0.053512	0.053769	0.075860	0.044259
4-D Model	0.061972	0.086907	0.106740	0.0743459

For the behavior evaluation 3 test course, the results are presented in table 12.

Table 12: 1-D model vs 4-D model for BE3

	$\dot{u}$	$\dot{v}$	$\dot{w}$	$\dot{r}$
1-D Model	0.051806	0.030311	0.060022	0.047931
4-D Model	0.058105	0.052425	0.078260	0.057432

A deeper analysis demonstrated that for the MSE for 1-D model was better than the 4-D model. We proceeded with the outputs from the 1-D model for our EKF algorithm.

4.8 NEURAL NETWORK MODEL VS DYNAMIC MODEL

The neural network architecture outputs were compared to the dynamic model outputs. The results for the three behavior evaluation test courses were presented in figures 16 to 18.

4.8.1 BEHAVIOR EVALUATION 1

Figure 16 demonstrates the path predicted by the dynamic filter and the neural network filter for the behavior evaluation test course 1.

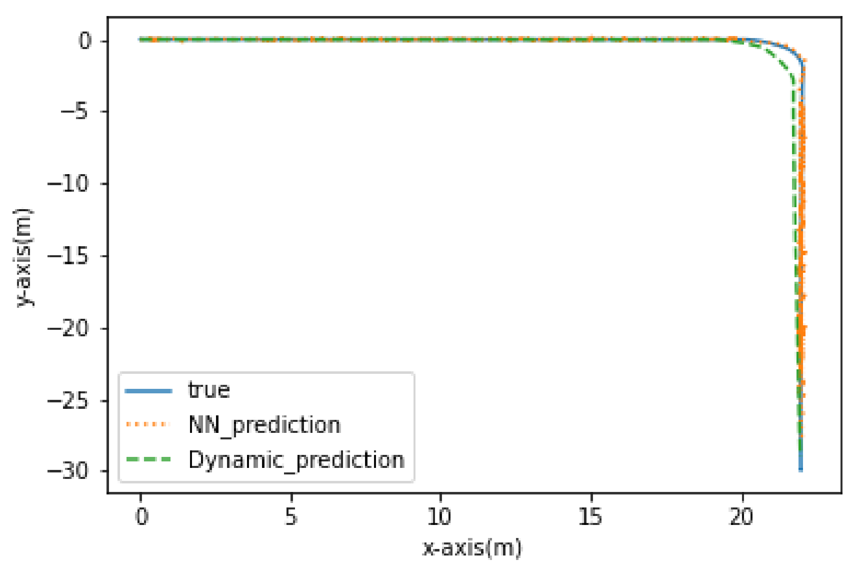

Figure 18: Behavior Evaluation 1

As seen in figure 18, the neural network filter predictions were closer to the truth value. However, on further analysis using MSE values (m), it was observed that the neural network had a few anomalies resulting in larger error compared to the dynamic filter.

Table 13: Results of Behavior Evaluation 1

Error (m)	Dynamic Filter	Neural Network Filter
Total	0.7572	1.5414
X-Kalman	0.5712	0.4148
Y-Kalman	0.4971	1.4846

From table 13, the total error of the dynamic filter was less than the neural network filter. The neural network needs to be improvement in order to make better predictions for the Y-Kalman values. In order to make further assessment of the filters, they were tested on the behavior evaluation 2 test course.

4.8.2 BEHAVIOR EVALUATION 2

Figure 17 shows the path predicted by the dynamic filter and the neural network filter.

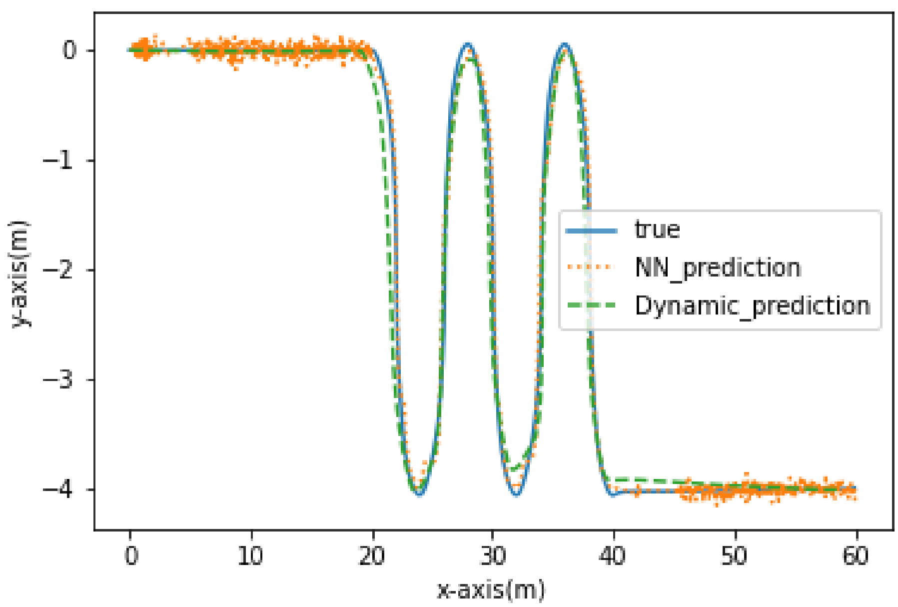

Figure 19 : Behavior Evaluation 2

As seen in figure 19, the neural network filter was closer to the true value compared to the dynamic filter. The dynamic filter converged faster to the ground truth value while the neural network values kept fluctuating on the y-axis values as seen at the beginning and end of figure 17.

This further supports our conclusion from table 14 that the neural network should be improved to make better predictions on the y-axis.

Table 14 : Results of Behavior Evaluation 2

Error (m)	Dynamic Filter	Neural Network Filter
Total	0.6697	0.6254
X-Kalman	0.3419	0.2391
Y-Kalman	0.5759	0.5779

From table 14, the mean error of the neural network filter was almost equal to the dynamic filter. Both filters can handle the simultaneous changes in the x-axis and y-axis.

4.8.3 BEHAVIOR EVALUATION 3

The third test course was a U-shaped path with clockwise and counterclockwise curves and straightaways.

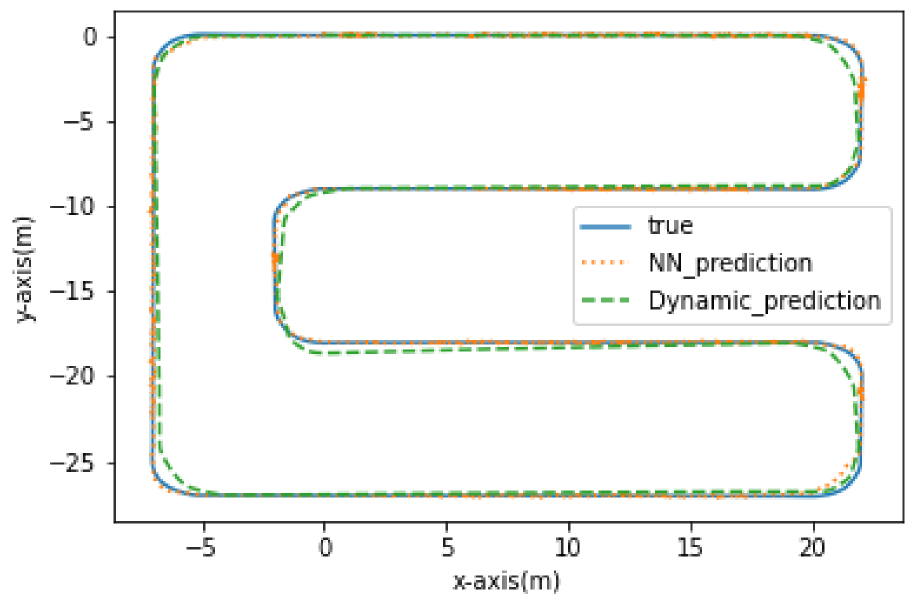

Figure 20: Behavior Evaluation 3

From figure 20, in behavior evaluation course 3, which includes all possible movements of the ROV, the dynamic filter and neural network filter exhibit a similar performance.

Table 15 : Results of Behavior Evaluation 3

Error (m)	Dynamic Filter	Neural Network Filter
Total	0.7629	0.7974
X-Kalman	0.5799	0.4557
Y-Kalman	0.4958	0.6544

On further analysis, the results of the neural networks discussed above were not replicable since the learning depended on the initial initialization of the weights. The performance of the networks was almost the same as discussed but a fluctuation of the order of $10^{-2}\ m/s^2$ was observed for the predicted acceleration values.

CONCLUSION

A simplified 4 DoF dynamic model was derived for a RexROV and it was implemented on an EKF. The outputs from the EKF were fed to a pure pursuit controller to control the RexROV in the simulation. From the figures 8 to 10 and the tables 5 to 7, the dynamic Kalman filter predictions were very close to the ground truth values. On comparison of the individual values predicted by the dynamic filter and the kinematic filter with the actual position of RexROV, the dynamic filter was more stable and reliable than the kinematic filter.

Since the dynamic model required the use of many coefficients, a neural network-based approach was used to model the dynamics of the underwater vehicle. Several iterations, involving four gradient descent algorithms and seven nonlinear activation functions, were performed to choose the best architecture for the neural network. It was found that AdaGrad gradient descent algorithm when used in combination with Scaled ELU activation function gave better results that the other combinations. A comparison study was conducted on the three behavior test courses to choose between the four 1-D models and 4-D model. It was found that the four 1-D models were able to predict the acceleration values much more accurately than a single 4-D model. The networks were trained to the maximum level without overfitting. The acceleration in y-axis($\dot{v}$) and the acceleration in x-axis ($\dot{u}$) were predicted to an accuracy of $0.05\ m/s^2$ on a random test course.

The 1-D neural network models were used as an input to the EKF algorithm and the results were compared to the dynamic filter. Both the neural-network and dynamic model were implemented as a post-processing tool to maintain uniformity. The accuracy of the neural network model was lower for the y coordinates on behavior evaluation test course 1. On further analysis, it was found that the network was able to make better prediction on behavior evaluation test courses 2 and 3. The neural network was able to sufficiently replicate the dynamics of the vehicle although a few more regularization and tuning needed to be performed to improve the network.

The neural network model was implemented as a post-processing tool and needs to be implemented in the simulation in real-time to verify the performance. The neural network codes were written in Pytorch and the controller codes on UUVSim used Cplusplus. Integrating the Pytorch and Cplusplus environment should be done before the neural network model can be run in real-time in UUVSim. The results were based on movement only in the two primary axes, x and y, similar to a ground vehicle. Further research is being conducted on the simultaneous triaxial movement to make the controller universal. It was expected that the movement on the z-axis should not affect the performance of the dynamic Kalman filter and neural network filter.

Currently, the model uses GPS data to correct position estimation. However, research has indicated that the strength of the electromagnetic waves for the GPS signal reduces significantly underwater [27]. Methods such as station keeping, SONARSLAM and vision systems were currently being explored as an alternative for position estimation. The results presented in this thesis were obtained at a constant wave velocity. The wave velocity can be used as an input to the neural network to better model the dynamics. The predictions were conducted at a frequency of 10Hz and the dynamic filter worked on the simulation without any performance issues. However, the filters need to be implemented in real-time hardware to assess the actual computational performance.

Another field of interest for the AUVSL group is prediction of the position of the vehicle directly without the use of filters. Recurrent Neural Network algorithms such as LSTM can be used to serve this purpose. One way to improve the Kalman filter predictions was to model the Q and R matrix accurately. Generative Adversarial Nets (GANs) can be used to improve the accuracy for Q and R. However, the GANs were limited by the availability of ground truth data for training.

The neural network modelling techniques discussed here were also applicable to the ground vehicle which is currently explored by other members of AUVSL group. Alternative controller techniques, such as sliding mode controller, were being researched to improve the performance of the model.

REFERENCES

[1] Stillman, D. *Oceans : The Great Unknown*. Ed. by National Aeronautics and Space Administration. 2009. url: https://www.nasa.gov/audience/forstudents/5-8/features/oceans-the-great-unknown-58.html (visited on 06/29/2019).

[2] Manhaes, M.M.M.,et al.“*UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation*”. In : *OCEANS 2016 MTS/IEEE Monterey*.*IEEE,Sept.2016*.doi:10.1109/oceans.2016.7761080.

[3] Kermorgant, O., “A Dynamic Simulator for Underwater Vehicle-Manipulators”. In: vol. 8810. Oct. 2014.doi:10.1007/978-3-319-11900-7_3.

[4] Kalman, R.E., “A New Approach to Linear Filtering and Pre-diction Problems”. In : *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[5] Julier, S. J. and Uhlmann, J. K. "Unscented filtering and nonlinear estimation," in *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401-422, March 2004, doi: 10.1109/JPROC.2003.823141.

[6] Romagos, D., Ridao, P., and Neira, J. *Underwater SLAM for Structured Environments Using an Imaging Sonar*. Vol.65. Jan.2010. isbn:978-3-642-14039-6.doi:10.1007/978-3-642-14040-2.

[7] Hascaryo, R. W., 2019. “*Reduced order extended Kalman filter incorporating dynamics of an autonomous underwater vehicle for motion prediction*”, Master’s Thesis, Department of Aerospace Engineering, University of Illinois at Urbana Champaign, USA.

[8] Berg, V., 2012. “*Development and commissioning of adp system for rov sf 30k*”, Master’s thesis, Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway, 2012.

[9] Adam, A., and Erik, E., 2016. “*Model-based design, development, and control of an underwater vehicle*”, Master’s thesis, Department of Electrical Engineering, Linkoping University, Linkoping, Sweden, 2016.

[10] Govinda, L., Tomas, S., Bandala, M., Nava Balanzar, L., Hernandez-Alvarado, R., and Antonio, J., 2014.“*Modelling, design and robust control of a remotely operated underwater vehicle*”. International Journal of Advanced Robotic Systems,11, 01, p. 1.

[11] Norris, W.R., and Patterson, A.E., 2019. “*System-level testing and evaluation plan for field robots: A tutorial with test course layouts*”. Robotics, 8(4).

[12] Abadi, M., Agarwal, A., Barham, P., et al *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from tensorflow.org.

[13] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d extquotesingle Alch'e-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[14] Fossen, T. I., *Handbook of Marine Craft Hydrodynamics and Motion Control*, 1st Ed., Wiley, Chichester, United Kingdom, 2011, Chapters 1-7, 12.

[15] Fossen, T. I., "*Nonlinear Modelling and Control of Underwater Vehicle*", PhD dissertation, Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway, 1991.

[16] Rosa, R., Zaffari, G., Evald, P., Drews-Jr, P., Botelho, S. (2017). "*Towards Comparison of Kalman Filter Methods for Localisation in Underwater Environments"*, 10.1109/SBR-LARS-R.2017.8215339.

[17] Manhaes, M., Scherer, S. A., Douat, L. R., Voss, M. and Rauschenbach, T. "*Use of simulation-based performance metrics on the evaluation of dynamic positioning controllers*", OCEANS 2017 - Aberdeen, Aberdeen, 2017, pp. 1-8,doi: 10.1109/OCEANSE.2017.8084658.

[18] Kleene, S.C. (1956). "*Representation of Events in Nerve Nets and Finite Automata*". Annals of Mathematics Studies (34). Princeton University Press. pp. 3–41. Retrieved 17 June 2017

[19] Schmidhuber, J. "*Deep Learning in Neural Networks: An Overview*".In : *CoRRabs/1404.7828* (2014). arXiv:1404.7828. url : http://arxiv.org/abs/1404.7828.

[20] Yiu, T., "*Understanding Neural Networks. Ed. by Towards Datascience",* 2019. url : https://towardsdatascience.com/understanding-neural-networks-19020b758230 (visited on 06/29/2019).

[21] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

[22] Duchi, J., Hazan, E., and Singer, Y., 2011. "*Adaptive Subgradient Methods for Online Learning and Stochastic Optimization"*. J. Mach. Learn. Res., 12:2121–2159.

[23] Hinton, G. 2012. Neural Networks for Machine Learning - Lecture 6a - Overview of mini-batch gradient descent.

[24] Kingma, D.P., and Ba, J. 2014. "*Adam: A Method for Stochastic Optimization*". CoRR, abs/1412.6980.

[25] Coulter, R. C., 1992. "*Implementation of pure pursuit path tracking algorithm*". The Robotics Institute, Carnegie Mellon University.

[26] Agudelo, J. G., 2015. "*Contribution to the model and navigation control of an autonomous underwater vehicle*", PhD thesis, PhD thesis, Departament d'Enginyeria Electronica, Universitat Politecnica de Catalunya.

[27] Taraldsen, G., Reinen, T. A., and Berg, T., 2011. *"The underwater gps problem"*, In OCEANS 2011 IEEE - Spain, pp. 1–8.