MARINE VEHICLE CHARACTERIZATION AND IMPLEMENTING VARIOUS LEVELS OF AUTONOMY

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE
UNIVERSITY OF HAWAI'I AT MĀNOA

IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

MECHANICAL ENGINEERING

DECEMBER 2024

By

Patrick Julian Ng

Thesis Committee:

Michael Krieg, Chairperson
John Allen
George Wilkens

Acknowledgments

Thank you to my thesis advisor Associate Professor Michael Krieg; his guidance and tutelage has been essential for my growth in academia. Thank you to Professor John Allen for mentoring me since senior year of my undergraduate work and helping me navigate graduate school. Thank you to Professor George Wilkens for teaching me statistics and reinforcing my appreciation of the exactitude of math. Thanks to the University of Hawai'i at Mānoa, the Mechanical Engineering Department, and the Ocean and Resources Engineering Department for providing me the opportunities to learn. Thanks to my colleagues in the Surface and Underwater Propulsion and Robotics Lab for their positivity and insight. Thanks to the National Science Foundation for funding this research. Thank you to my family for always believing in me.

Abstract

Remotely operated vehicles (ROVs) are marine submersible robots that serve a variety of purposes in industry and research. These unmanned vessels harness human judgment and precision to perform tasks within extreme environments like the deep sea, polar, and volcanic regions of the ocean. Some examples of their usages are to survey the ocean floor, maintain pipelines and collect scientific data in the form of sediment and hydrothermal vent plume samples and optical observations of marine wildlife. Training of ROV pilots is typically very expensive and time-consuming because of the highly specialized skill requirements. A novel system was proposed by collaborators from the University of Florida (UF) and the University of Hawaiʻi at Mānoa (UHM) for piloting ROVs with an intuitive augmented/virtual reality (AR/VR) interface that uses a hybrid autopilot. To demonstrate the feasibility of this system the group at UHM altered the ArduSub firmware of the commercially available BlueROV2 (BROV2) Heavy to enable model-based quantitative control. Error feedback control for the hybrid autopilot was implemented using Robot Operating System (ROS) in a modular manner that enabled various levels of autonomy to assist ROV pilots. Alongside the development of the custom firmware and hybrid autopilot, the software-in-the-loop (SITL) simulation environment was also updated with an experimentally determined hydrodynamic model using onboard sensor-based system identification techniques As much as sixty percent improvement of relative error when predicting vehicle behavior compared to the original SITL model. The calibration process of the hybrid autopilot involved iteratively cycling between SITL and water tank testing. It was demonstrated that this procedure was an effective method for achieving precision control of the BROV2.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Unmanned Underwater Vehicles (UUVs) are a class of marine robot that are deployed without a pilot physically present onboard. There are two primary types of UUV, the remotely operated vehicle (ROV) which has a pilot controlling the vehicle from some remote location, and the autonomous underwater vehicle (AUV) which has no pilot in the loop. Historically, the first use of ROVs in the 1960s predated the rise of AUVs in the 1990s; ROVs were rapidly developed to serve the interests of the oil and gas industries [2]. These unmanned vehicles assist in the exploration of extreme environments that would be dangerous for direct human involvement.

Autonomous underwater vehicles are recognizable by their torpedo-shaped hullform (Figure 1.1), as they are typically designed to travel long distances underwater efficiently while minimizing drag. Autonomous underwater vehicles have evolved a wide range of applications over the years, they have garnered interest for covert defense applications scouting ahead of battle and responding to terroristic threats to ports [3]. They are also well-suited for oceanographic science applications when equipped with the appropriate sensors. Finding extensive use in marine geoscience, tracking hydrothermal vent fluids, exploring extreme polar environments, and able to do so at great depths [4]. Using acoustic-based sensors AUVs can simultaneously map bathymetry and navigate by tracking the relative motion of the seafloor [5]. Utilization of multiple AUVs in a fleet for monitoring spaces in the ocean, search and rescue operations, and seismic activity tracking is discussed in [6]. For a more comprehensive list of purposes and examples of AUVs refer to [1].

Unlike AUVs, which have no human intervention, ROVs have pilots that can leverage human judgment and discretion for close-range missions requiring more nuanced control. Remotely operated vehicles continue to be widely used for maintenance, surveying, and monitoring in the oil/gas and

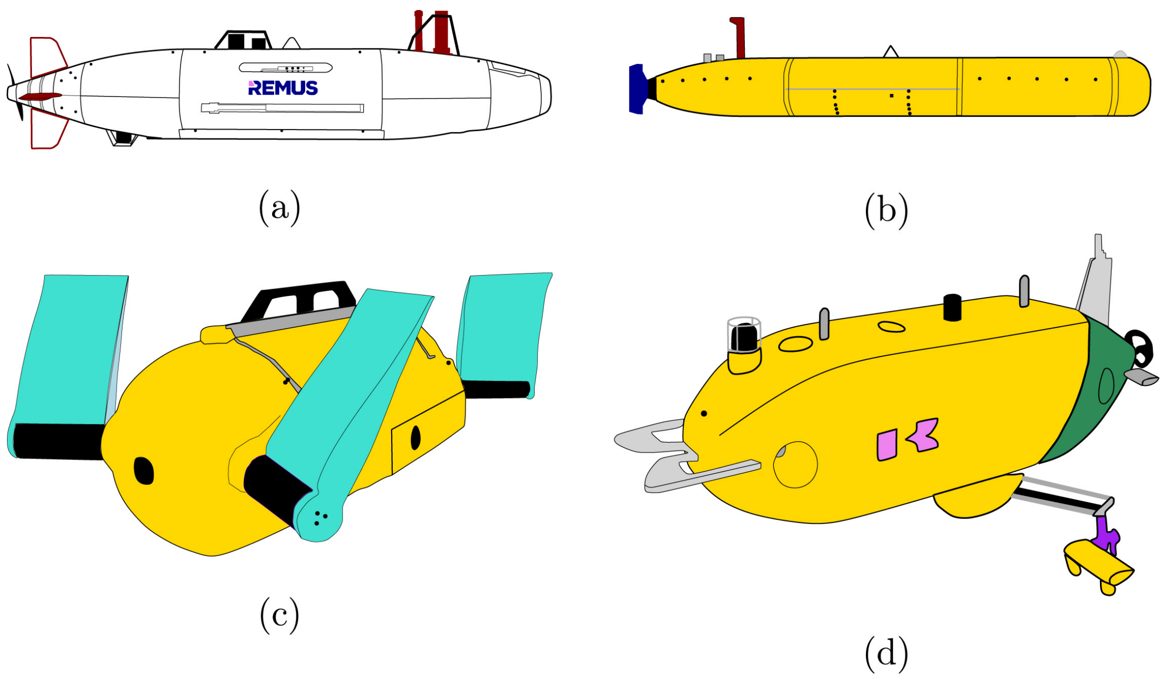


Figure 1.1: Some illustrations of autonomous underwater vehicles (AUVs) (a) REMUS-6000, (b) Bluefin-21, (c) U-CAT, and (d) Kawasaki AUV, citations for each individual vehicle are in [1].

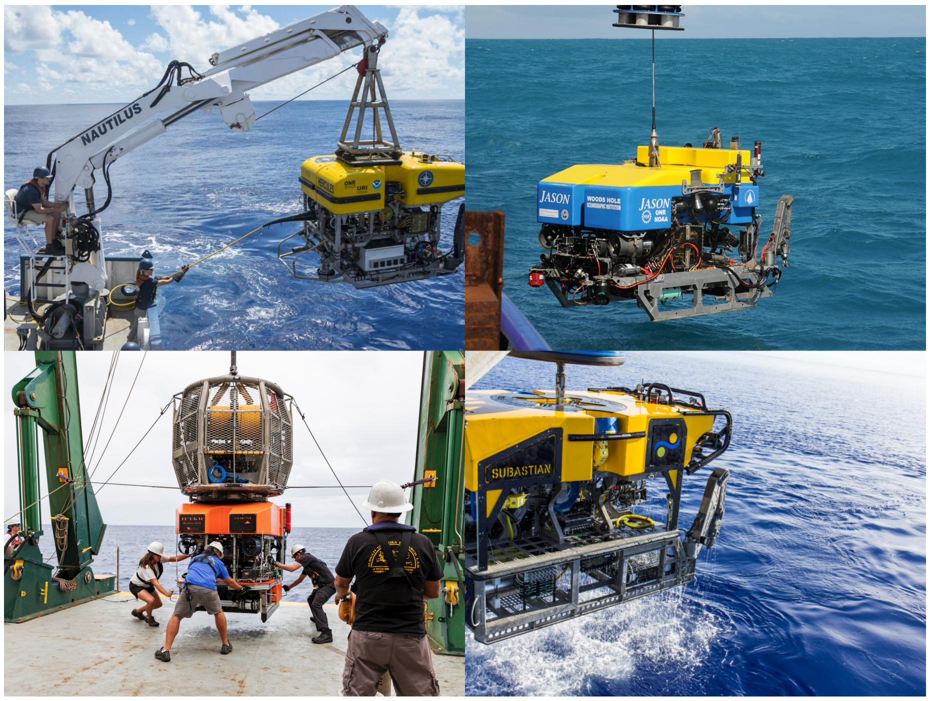


Figure 1.2: Clockwise from top left: Hercules [7], Jason [8], SuBastian [9], and Lu'ukai [10] are examples of research remotely operated vehicles (ROVs) currently in service.

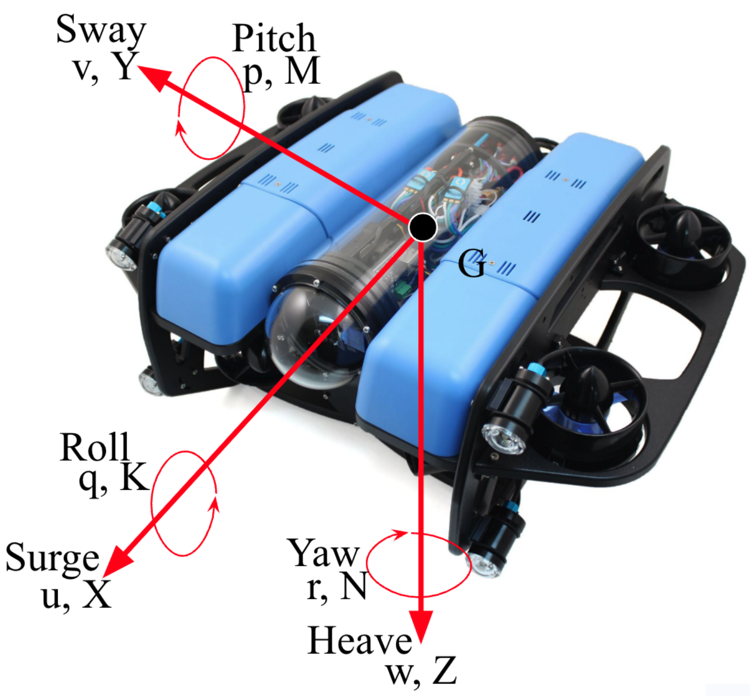


Figure 1.3: The BlueROV2 (BROV2) Heavy shown from an angled perspective annotated with body-fixed reference frame symbols and nomenclature. The body-fixed coordinate axes are attached to the vehicle's center of mass G. In the Heavy configuration it has eight thrusters; four are positioned vertically on the upper layer and four are positioned at symmetric angles on a lower plane not totally visible in the figure.

other industries. Rare and valuable minerals previously not economically feasible to collect has had growing interest, in January 2024 Norway became the first country to allow deep-sea mining in its waters [11]. The Scientific and Environmental ROV Partnership using Existing Industrial Technology (SERPENT) Project, leverages industrial ROVs regular access to deep-sea environments in sharing scientific data with the research community [12]. They are also used extensively in marine geoscience to sample hydrothermal vents and collect sediment samples with manipulator arms [13, 14]; monitor the movements of fish assemblages with vision-based sensors [15, 16]; as well as other aquaculture and oceanographic purposes. Some examples of ROVs used for research are shown in Figure 1.2.

As technology becomes more affordable ROVs have become increasingly commonplace in a variety of settings. Reflecting ROVs' rise in ubiquity is children's positive perception of them [17] and growing interest in building custom ROVs from scratch [18, 19, 20, 21].

There have been a few startups that have previously offered premade consumer-priced ROVs like Trident ROV by OpenROV. However, not many of these startups have stood the test of time, and in the case of OpenROV, was bought by a larger company Sofar Ocean [22] that have a more industrial-centric customer base. One company that has startup origins and continues to grow is Blue Robotics, they have established themselves as a prominent source of affordable high-quality marine robotics equipment like thrusters, tethers, and pressure vessels. Their flagship product is the BlueROV2 (BROV2) [23], which comes in standard six-thruster or eight-thruster Heavy configurations (Figure 1.3). This vehicle has been popular among researchers because it is built on an open-source platform that facilitates software, hardware, and firmware customization. It has been used in aquaculture to track the length of kelp with machine learning image processing [24] and been equipped with a DVL, sonar, and stereo vision camera for under-ice sensing [25]. Such projects greatly benefit from the hacking ethos of the BROV2, where other commercially available vehicles actively discourage user modifications.

The skills required of ROV pilots are highly specialized and have historically required extensive time-consuming training resulting in a limited workforce that ultimately hinders applications of ROVs. In a joint venture between the University of Florida (UF) and the University of Hawaii at Manoa (UHM) a novel framework was proposed that addresses this issue with an augmented/virtual reality (AR/VR) interface for pilots [26, 27]. There are five modules to this project: ROV, Subsea Sensing, Workplace Model, Robotic Simulation and Control, and Human Operator. These modules together provide an interface for human pilots to control ROVs from remote locations as far as Florida to Hawaii with immersive AR/VR that accelerates training and improves accessibility of the professional workforces. The purview of our research group at UHM is the ROV and Robotic Control modules. The ROV used is the BlueROV2 (BROV2) Heavy in Hawaii that the AR/VR pilot in Florida controls remotely. The Robotic Control module includes how the AR/VR gestures are mapped to BROV2 movements and other autopilot functions that assist the remote pilot.

In order to design an effective autopilot, accurate system identification of the BROV2 is necessary. Typical methods employ tow tank testing [28] or motion capture systems [29] to record dynamics of a system from which models are derived, these are specialized tools which are prohibitively expensive if facilities are not already in place. Procedures for modeling a marine vehicle using onboard sensors [30, 31] instead, shall be proven to be a reliable alternative. In this thesis, the governing equations for the BROV2 and the base model for the autopilot shall be introduced first. The physical model description of the BROV2 Heavy and actuators are reviewed in the following chapter. Experimental open-loop water tank testing that determined the hydrodynamic coefficients for the yaw and heave degrees of freedom shall be discussed next. These models go on to update the open-source simulation environment to produce more accurate results. The last chapter will cover implementation details of the autopilot including simulation testing and experimental validation of calibrated control laws. Autopilot features are designed to be mixed-and-matched in Robot Operating System (ROS) depending on use case for various levels of autonomy. Earlier work documented in [32] will also be supporting findings. This thesis shall demonstrate methodologies for building robust quantitative control for the BROV2 Heavy which can be replicated for other payload configurations.

Chapter 2

Conventions and Models

In this chapter mathematical conventions, coordinate systems, and symbols used in this thesis shall be defined. Once conventions and coordinate systems are established, the full six degrees of freedom (6DOF) governing equations model for the BROV2 shall be presented. The particular hydrodynamic model used throughout the rest of this work shall also be specified.

2.1 Coordinate Systems and Conventions

In this thesis, bolded variables represent vectors or matrices and non-bolded variables represent scalars. The coordinate systems used will have three orthogonal axes. For any given coordinate system let the first, second, and third axes be defined with the generic frame-agnostic unit vectors $\hat{\boldsymbol{\imath}}$, $\hat{\boldsymbol{\jmath}}$, and $\hat{\boldsymbol{k}}$, respectively. Thus by the right-hand convention, the cross product of $\hat{\boldsymbol{\imath}}$ and $\hat{\boldsymbol{\jmath}}$ generates $\hat{\boldsymbol{k}}$. [33] When viewing a unit vector such that it is pointing out of the page, the positive direction of rotation about the axis is counter-clockwise. As such, every linear degree of freedom has an angular degrees of freedom associated with it.

2.1.1 Inertial and Body-fixed Reference Frames

There are two primary coordinate systems that shall be referenced throughout this work: the inertial and body-fixed frames. Together, these two reference frames define an object's geometric state. Adopting the SNAME conventions [34] the linear degrees of freedom in the inertial frame shall be referred to as

$$\boldsymbol{\eta_1} = [x, y, z]^\top . \tag{2.1}$$

Position/Orientation in Inertial Frame		
Symbol	Vector	
x	$\boldsymbol{\eta_1}$	$\boldsymbol{\eta}$
y		
z		
ϕ	$\boldsymbol{\eta_2}$	
θ		
ψ		

Linear/Angular Velocities in Body-Fixed Frame			
Direction	Symbol	Vector	
Surge	u	$\boldsymbol{\nu_1}$	$\boldsymbol{\nu}$
Sway	v		
Heave	w		
Roll	p	$\boldsymbol{\nu_2}$	
Pitch	q		
Yaw	r		

External Forces/Moments in Body-Fixed Frame			
Direction	Symbol	Vector	
Surge	X	$\boldsymbol{\tau_1}$	$\boldsymbol{\tau}$
Sway	Y		
Heave	Z		
Roll	K	$\boldsymbol{\tau_2}$	
Pitch	M		
Yaw	N		

Table 2.1: Throughout this thesis these symbols and conventions are used for the 6DOF marine vehicle parameters.

Angular degrees of freedom are contained in

$$\boldsymbol{\eta_2} = [\phi, \theta, \psi]^\top , \tag{2.2}$$

where the Euler angles ϕ, θ, and ψ describes the orientation of the body-fixed frame with respect to the inertial frame. The complete 6DOF representation of position in the inertial frame is

$$\boldsymbol{\eta} = [\boldsymbol{\eta_1}, \boldsymbol{\eta_2}]^\top = [x, y, z, \phi, \theta, \psi]^\top . \tag{2.3}$$

The body-fixed coordinate system is attached to the object's center of mass and aligned with the designated orientation (see the following Section 2.1.2). Also in line with SNAME conventions, body-fixed linear velocities shall be defined with

$$\boldsymbol{\nu_1} = [u, v, w]^\top . \tag{2.4}$$

Angular velocities in the body-fixed frame are

$$\boldsymbol{\nu_2} = [p, q, r]^\top . \tag{2.5}$$

The complete 6DOF representation of velocity in the body-fixed frame is

$$\boldsymbol{\nu} = [\boldsymbol{\nu_1}, \boldsymbol{\nu_2}]^\top , = [u, v, w, p, q, r]^\top . \tag{2.6}$$

External forces in the body-fixed frame are

$$\boldsymbol{\tau_1} = [X, Y, Z]^\top . \tag{2.7}$$

External torques in the body-fixed frame are

$$\boldsymbol{\tau_2} = [K, M, N]^\top . \tag{2.8}$$

The complete 6DOF representation of forces and torques in the body-fixed frame is

$$\boldsymbol{\tau} = [X, Y, Z, K, M, N]^\top . \tag{2.9}$$

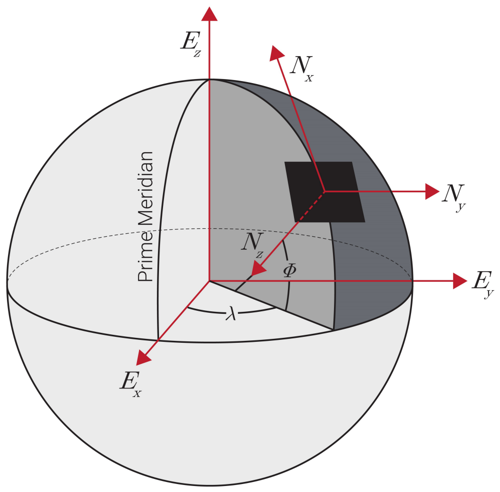
These 6DOF motions in the body-fixed frame are named surge, sway, heave, roll, pitch, and yaw. Figure 1.3 shows how the body-fixed frame variables and names apply to the BROV2. There is no standard terminology for linear motions in the inertial frame; the terms “roll,” “pitch,” and “yaw” are also used to describe the Euler angles ϕ, θ, and ψ in Equation (2.2).

Because the body-fixed reference frame moves and rotates with the object, position and orientation values of the object itself in the body-fixed reference frame are always zero. Said differently, the origin of the body-fixed coordinate system coincides with the body’s center of gravity and the axes of the body-fixed reference frame are rigidly aligned with the chosen coordinate axes, i.e. Forward, Starboard, Down or Forward, Port, Up of the body (see the next section for more details). Instances where objects or quantities known in the body-fixed frame shall be indicated with a “b” subscript.

2.1.2 Choice of Reference Frames

The inertial frame is a fixed coordinate system aligned with the cardinal directions either the North-East-Down (NED) or East-North-Up (ENU), where “down” is defined as the direction of gravity. These acronyms name the first and second axes in order. For ENU, the first axis points East and the second points North, causing the third to point away from the center of the earth.

The body-fixed reference frame is a dynamic coordinate system that moves and rotates with an object. Forward-Right-Down/Forward-Starboard-Down (FRD/FSD) and Forward-Left-



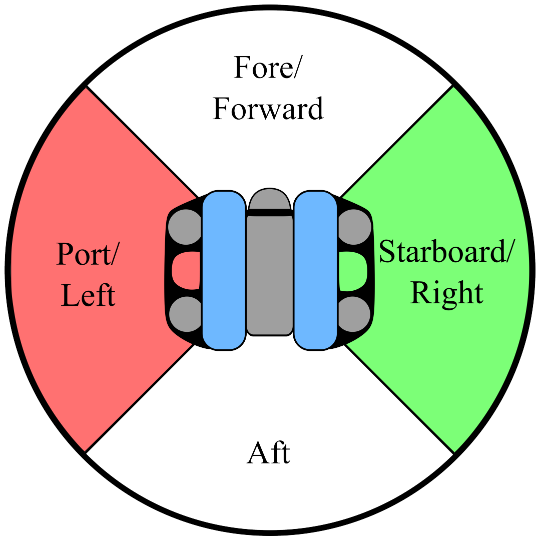


Figure 2.1: An example of a North-East-Down inertial reference frame labeled with N is shown on the left [33]. An illustration of the body-fixed frame directions of the BROV2 is shown on the right. Body-fixed directions are labeled as the fore/forward (F) direction is in the direction of the pressure vessel dome, port/left (P/L) points from the center towards the green region, starboard/right (S/R) is the red region, and the back end is referred to as aft. Not shown is the up direction that points out of the page out the top of the vehicle, and the down direction that points into the page out the bottom of the vehicle.

Up/Forward-Port-Up (FLU/FPU) body-fixed reference frames are paired with the analogous inertial reference frame, i.e. NED-FSD or ENU-FPU (Figure 2.1).

The open-source firmware used by the BROV2 is known as ArduSub, in this work it was updated with custom nonstandard features to enable quantitative control of the BROV2. Most of the ArduSub codebase is written for NED-FSD, so throughout this thesis derivations relating to the firmware will be presented in NED-FSD reference frames. However, both the sensor data from and control of the BROV2 were handled by Robot Operating System (see Appendix D) which follows a strict ENU-FPU standard. As such, graphs of sensor data shall be presented in the ENU-FPU frames. Any exceptions shall be made clear.

2.1.3 Converting Between Reference Frames

The inertial frame coordinate system is an earth-fixed reference frame, aligned with the ENU/NED directions, that is centered on the latitude, longitude, and altitude values at robot initialization. The body-fixed coordinate system is a moving reference frame, aligned with the FSD/FPU axes, attached to the object's center of gravity. The bridge between the two reference frames is the inertial frame angular pose, $\boldsymbol{\eta_2}$ that expresses the Euler angles that the body-fixed axes are rotated with respect to the static inertial reference axes. Frequently, some calculations are best suited to be performed in one frame and need to be output to another, in this section the transformation matrices used to do so are discussed.

Orthonormal rotation matrices are used to transform linear vectors from one frame to the other. A sequence of rotations can be expressed mathematically with matrix multiplication. The product of any number of orthonormal matrices is another orthonormal matrix. These matrices use angular states contained in $\boldsymbol{\eta_2}$ to rotate the coordinate axes without stretching or shrinking. Given any angular orientation in space, there are an infinite amount of combinations of rotations that could be performed to arrive at that state from a non-rotated starting point. Generally, three rotations are sufficient, one about each axis. In this work the Tait-Bryan Euler convention (zyx) shall be used.

The resultant zyx rotation matrix that converts body-fixed linear quantities to the inertial frame is

$$\boldsymbol{J_{1,\eta_2}} = \boldsymbol{J_1}(\boldsymbol{\eta_2}) = \begin{bmatrix} c(\psi)c(\theta) & c(\psi)s(\theta)s(\phi) - s(\psi)c(\phi) & c(\psi)s(\theta)c(\phi) + s(\psi)s(\phi) \\ s(\psi)c(\theta) & s(\psi)s(\theta)s(\phi) + c(\psi)c(\phi) & s(\psi)s(\theta)c(\phi) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix}, \tag{2.10}$$

where $c(\cdot)$ is the cosine function and $s(\cdot)$ is the sine function. Converting from inertial frame linear coordinates to body-fixed coordinates uses the inverse of $\boldsymbol{J_1}(\boldsymbol{\eta_2})$ which for orthonormal rotation matrices is

$$\boldsymbol{J_1}^{-1}(\boldsymbol{\eta_2}) = \boldsymbol{J_1}(\boldsymbol{\eta_2})^\top. \tag{2.11}$$

Transformation matrices for zero-order angular orientation and angular velocities are not orthonormal. This stems from the fact that $\boldsymbol{J_1}^{-1}(\boldsymbol{\eta_2})$ is a series of rotations and the inherent nonlinearity of angles. The resulting non-orthonormal transformation matrix to convert body-fixed angular velocities and other angular quantities in the body-fixed frame (e.g. proportional errors) to the inertial frame is the matrix

$$\boldsymbol{J_2}(\boldsymbol{\eta_2}) = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi)/c(\theta) & c(\phi)/c(\theta) \end{bmatrix} \tag{2.12}$$

and its inverse for converting inertial frame angular velocities to body-fixed angular velocities is

$$\boldsymbol{J_2}^{-1}(\boldsymbol{\eta_2}) = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & c(\phi) & s(\phi)c(\theta) \\ 0 & -s(\phi) & c(\phi)c(\theta) \end{bmatrix}, \tag{2.13}$$

where $t(\cdot)$ is the tangent function.

2.2 Hydrodynamic Model of the BlueROV2

In the body frame of the vehicle the governing dynamics experienced can be expressed as

$$\boldsymbol{\tau} = (\boldsymbol{M} + \boldsymbol{M_A})\,\dot{\boldsymbol{\nu}} + (\boldsymbol{B_L} + \boldsymbol{B_{NL}}(\boldsymbol{\nu}) + \boldsymbol{C}(\boldsymbol{\nu}))\,\boldsymbol{\nu} + \boldsymbol{g}\,(\boldsymbol{\eta_2})\,, \tag{2.14}$$

where $\boldsymbol{\tau}$ is the 6DOF external forces and torques, $\boldsymbol{M}$ is the system rigid body inertia matrix, $\boldsymbol{M_A}$ is the added mass matrix, $\dot{\boldsymbol{\nu}}$ are body frame accelerations, $\boldsymbol{B_L}$ is the linear damping matrix, $\boldsymbol{B_{NL}}(\boldsymbol{\nu})$ is the nonlinear damping matrix, $\boldsymbol{C}(\boldsymbol{\nu})$ is the Coriolis-centripetal matrix, where both $\boldsymbol{C}$ and $\boldsymbol{B_{NL}}$ are functions of the body frame velocities $\boldsymbol{\nu}$, and $\boldsymbol{g}(\boldsymbol{\eta_2})$ is the restoring forces matrix that is a function of the angular orientation of the vehicle $\boldsymbol{\eta_2}$. The model which Equation (2.14) is based on is the standard model proposed by Fossen [35]. Values for added mass can be approximated with analytical techniques like strip theory and potential flow [36], but these techniques cannot be used to calculate viscous forces. Hypothetically, drag can be solved analytically through an iterative process; whereby, potential flow is used to determine a surface velocity distribution, then boundary layer theory can be used to approximate local stresses by equating the free-stream velocity to the previously calculated potential flow velocity. The boundary layer thickness is then added to the body geometry and the whole process is repeated until the solution converges. However, this is an extremely cumbersome process. These analytical techniques can be implemented with computers. Given enough computing resources and time, added mass and drag can be determined with high fidelity computational fluid dynamic simulations. A practical alternative is experimentally developed empirical models that can also capture phenomena not predicted by analytical models. Forces and torques due to Coriolis-centripetal effects result from combined motions in two or more degrees of freedom. Elsewhere in this thesis only individual degrees of freedom will be modeled, causing quantities scaling with $\boldsymbol{C}$ drop to zero.

2.2.1 Inertial and Drag Forces

Matrices containing the rigid body coefficients in $\boldsymbol{M}$ and the hydrodynamic coefficients in $\boldsymbol{M_A}$, $\boldsymbol{B_L}$, and $\boldsymbol{B_{NL}}(\boldsymbol{\nu})$ shall be made of only diagonal terms. Such that, the system rigid body inertia matrix

$$\boldsymbol{M} = \boldsymbol{diag}(m, m, m, I_{xx}, I_{yy}, I_{zz}) \tag{2.15}$$

where m is mass of the vehicle and I_{jj} is the moment of inertia about the j-axis; the added mass matrix

$$\boldsymbol{M_A} = \boldsymbol{diag}(X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}, K_{\dot{p}}, M_{\dot{q}}, N_{\dot{r}}). \tag{2.16}$$

In general, the symbols of the hydrodynamic coefficients use capital letters corresponding to the body-frame forces/torques and subscripts referring to which quantities they scale with (see Table 2.1). Added mass coefficients scale linearly with body-frame accelerations. Linear damping terms are grouped in

$$\boldsymbol{B_L} = \boldsymbol{diag}(X_u, Y_v, Z_w, K_p, M_q, N_r), \tag{2.17}$$

and scale linearly with body-frame velocities. Nonlinear terms are grouped in

$$\boldsymbol{B_{NL}} = \boldsymbol{diag}(X_{u|u|}|u|, Y_{v|v|}|v|, Z_{w|w|}|w|, K_{p|p|}|p|, M_{q|q|}|q|, N_{r|r|}|r|), \tag{2.18}$$

notice how the nonlinear damping coefficients present are functions of the absolute values of the corresponding velocities.

The Coriolis and centripetal effects matrix

$$\boldsymbol{C}(\boldsymbol{\nu}) = \begin{bmatrix} \boldsymbol{0}_{3\times 3} & -\boldsymbol{m}\boldsymbol{S}(\boldsymbol{\nu}_1) \\ -\boldsymbol{m}\boldsymbol{S}(\boldsymbol{\nu}_1) & -\boldsymbol{S}(\boldsymbol{I}_0\boldsymbol{\nu}_2) \end{bmatrix} \tag{2.19}$$

is reduced to only centripetal terms because the body-fixed reference frame is centered on the vehicle's center of gravity. The total six-by-six matrix is constructed from four three-by-three submatrices. The upper left submatrix is made completely of zeros. The other submatrices use the skew-symmetric matrix to represent the cross produce of inertia term vectors with body-fixed velocities as a matrix-vector operation. In other words for the cross product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ the skew-symmetric matrix distributes terms in $\boldsymbol{a}$ into a matrix such that

$$\boldsymbol{S}(\boldsymbol{a})\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}. \tag{2.20}$$

The $\boldsymbol{I_0}$ term in the lower right submatrix is the lower right quadrant of the nonsimplified rigid body matrix (2.15) including nondiagonal terms, where

$$\boldsymbol{I}_0 = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{yx} & I_y & -I_{yz} \\ -I_{zx} & -I_{zy} & I_z \end{bmatrix}. \tag{2.21}$$

A more detailed model would include off-diagonal terms in the hydrodynamic coefficient matrices as well as the $\boldsymbol{C}$ matrix to describe forces due to coupled motions e.g. heave and surge or sway and roll. For this thesis, a simplified model using only diagonal matrices is used because a) the data collected from experiments described in later sections can only confirm those values, b) these terms have the primary influence on the vehicle, and c) one degree of freedom motions shall be analyzed.

2.2.2 Restoring Forces and Torques

The restoring forces and torques vector $\boldsymbol{g}(\boldsymbol{\eta_2})$ is purely a function of the vehicle's inertial orientation and volume. This model shall assume that the vehicle is completely submerged. Hydrostatic force due to buoyancy is a constant force resulting from the pressure caused by displaced water, always acting opposite of gravity. The expression of the force appears as either flotation, positive buoyancy, or sinking, negative buoyancy. Objects can be made of materials that produce a neutrally buoyant effect in which weight force and buoyancy force are in exact equilibrium, and remains motionless submerged in water. Let the force quantity

$$\boldsymbol{F_{B,net}} = \begin{bmatrix} 0 \\ 0 \\ W - B \end{bmatrix} \tag{2.22}$$

where $\boldsymbol{F_{B,net}}$ is in the inertial frame vector representing net buoyancy of a completely submerged vehicle, W is the weight scalar of the vehicle, and B is the buoyancy force scalar of the vehicle.

Net buoyancy's effect on the body depends on the object's orientation. Mathematically, this effect is the projection of the net buoyancy force onto the body-fixed coordinate axes

$$\boldsymbol{F_B} = \boldsymbol{J_{1,\eta_2}^{-1}} \boldsymbol{F_{B,net}} \tag{2.23}$$

where $\boldsymbol{F_B}$ is the three dimensional body-fixed net buoyancy force vector and $\boldsymbol{J_{1,\eta_2}^{-1}}$ is the inertial-to-body frame rotation matrix (2.11). Note that here and elsewhere, $\boldsymbol{F_B}$ is defined as the *body-fixed* net buoyancy force vector that is a function of $\boldsymbol{\eta_2}$, and $\boldsymbol{F_{B,net}}$ refers to the *inertial* net buoyancy vector that is constant

Given a submerged object, its center of buoyancy is defined as the location of the centroid of the vehicle's volume. The body-fixed vector that points from the center of gravity to the center of buoyancy is $\boldsymbol{r_B}$. The center of buoyancy is where $\boldsymbol{F_B}$ act on the object, due to the offset from the center of mass, there are also restoring torques. The body will come to equilibrium with the center of gravity directly above the center of buoyancy. If the center of buoyancy is not directly above the center of gravity the restoring torque will drive the object to its stable configuration, any other static position is very unstable. Figure 2.2 shows examples of an AUV pitched and rolled, the moment arm and buoyancy force creates a restoring torque that drives the vehicle to a stable orientation. Let the vector

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ B \end{bmatrix} \tag{2.24}$$

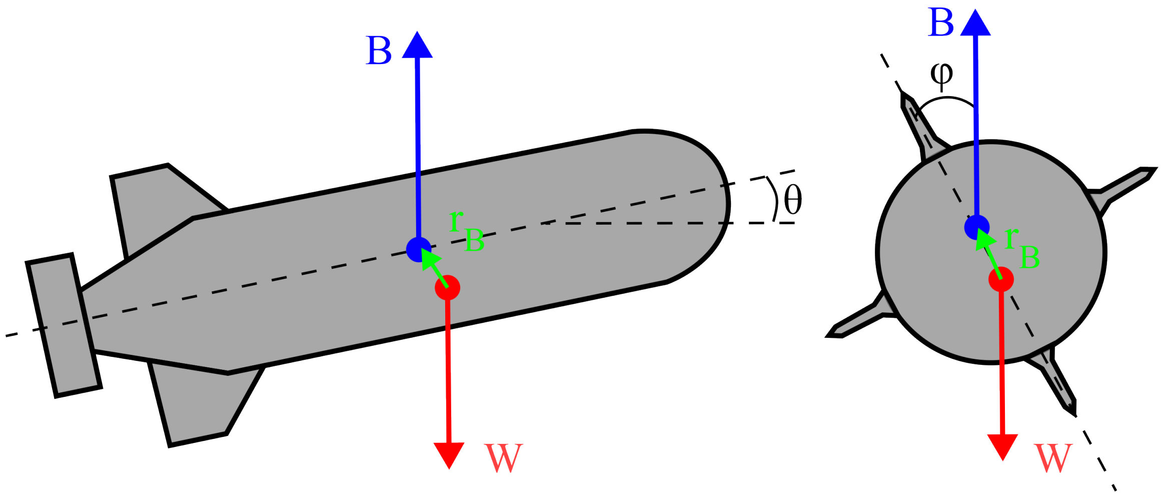


Figure 2.2: Illustrations of an AUV showing the buoyancy vector ($\boldsymbol{B}$) acting on the center of buoyancy, weight vector ($\boldsymbol{W}$) acting on the center of gravity, and the buoyancy moment arm vector ($\boldsymbol{r_B}$). The left illustration shows the AUV with a positive pitch, and the right shows the AUV with a positive roll.

be the inertial-frame buoyancy force. Since the body-fixed reference frame is attached to the vehicle's center of gravity, the restoring torques are only a function of the buoyancy force, not the net force involving weight. Therefore the vector of restoring torques acting on the body can be determined with

$$\boldsymbol{\tau_B} = \boldsymbol{r_B} \times \boldsymbol{J_{1,\eta_2}^{-1}} \boldsymbol{B} \tag{2.25}$$

where $\boldsymbol{\tau_B}$ is the three dimensional torque vector of restoring torques. Hence,

$$\boldsymbol{g}(\boldsymbol{\eta_2}) = \begin{bmatrix} \boldsymbol{F_B} \\ \boldsymbol{\tau_B} \end{bmatrix} \tag{2.26}$$

where $\boldsymbol{g}(\boldsymbol{\eta_2})$ is the 6DOF restoring forces and torques in the body-fixed frame, $\boldsymbol{F_B}$ are restoring forces as described in Equation (2.23), and $\boldsymbol{\tau_B}$ are restoring torques as described in Equation (2.25).

2.2.3 Piecewise Heave Model

For most degrees of freedom of the BROV2 the hydrodynamic coefficients can be approximated as equal in the positive and negative directions. It is symmetric across the x_b-z_b plane making sway properties equal. As well as port-starboard geometry making surge properties approximately equal. Moreover, it is nearly symmetric about the x_b-, y_b-, and z_b-axes making roll, pitch, and yaw properties in either rotational direction roughly equivalent, respectively. The one exception is the significant asymmetry about the x_b-y_b plane affecting the properties of heave. The bottom of the vehicle has complicated geometries due to the lateral thrusters, angular edges of the buoyancy fairings, and placement of frame. Whereas, the top of the vehicle has more smooth and rounded edges (Figure 2.3).

Experimental testing suggested that added mass also has different values given a positive or negative heave motion. The exact nature of the hydrodynamic model resulting from the asymmetry is unknown so the following assumption is made that the heave force can be fitted to a general form

$$Z = (m + Z_{\dot{w}})\dot{w} + Z_w w + Z_{w|w|} w^2 + \boldsymbol{F_B} \cdot \hat{\boldsymbol{k}} \tag{2.27}$$

where Z is the body heave force, m is the mass of the BROV2 Heavy, $Z_w \dot{w}$ is the added mass in heave, Z_w is the linear viscous damping in heave, $Z_{w|w|}$ is the quadratic damping in heave, w is the vehicle's heave velocity, $\dot{w}$ is the vehicle's heave acceleration, and $\boldsymbol{F_B} \cdot \hat{\boldsymbol{k}}$ is the vertical component of the body-frame restoring forces vector (2.23). The coefficients are defined by the piecewise functions

$$Z_{\dot{w}} = \begin{cases} Z_{\dot{w},rise} & \text{if } \dot{w} < 0, \\ Z_{\dot{w},dive} & \text{if } \dot{w} > 0, \end{cases} \quad Z_w = \begin{cases} Z_{w,rise} & \text{if } w < 0, \\ Z_{w,dive} & \text{if } w > 0, \end{cases} \quad Z_{w|w|} = \begin{cases} Z_{w|w|,rise} & \text{if } w < 0, \\ Z_{w|w|,dive} & \text{if } w > 0. \end{cases} \tag{2.28}$$

Here, each hydrodynamic coefficient switches between rise and dive values based on the directions of heave acceleration and velocity in the FSD frame.

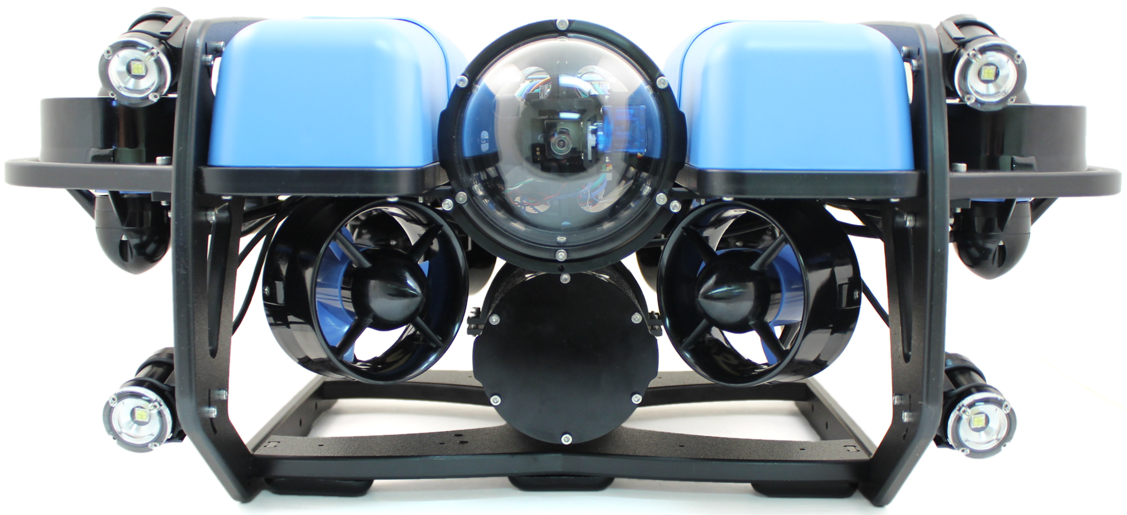


Figure 2.3: The front view of the BROV2 Heavy. It can clearly be seen that there is asymmetry between the top and bottom halves of the vehicle [23].

Chapter 3

Physical Model of the BROV2

In this chapter the models for how the eight thrusters effect the BROV2 are discussed. First the thruster allocation matrix is introduced and derived from the thruster layout of the vehicle. Next the inverse thruster allocation matrix implemented in the vehicle firmware so that motor forces can be solved for from desired forces and torques is presented. The firmware thruster model that uses a lookup table to solve for individual motor forces and signals is discussed last. Figure D.1 shows how these features connect to one another in the final control architecture.

3.1 Control Forces Applied Through Thruster Allocation Matrix

The derivations in this section look at actuation dynamics and how the vehicle actually implements a given control action. Calculations done by the firmware to accomplish these actions do not involve any pilot input. The coordinate frames of the ArduSub firmware follow the NED-FSD (Section 2.1.2) convention, and are presented as so. ROVs like the BROV2 owe their high maneuverability to the thrusters distributed around multiple locations on their boxy frame. Each of these thrusters work in tandem to impart control forces and torques on the body of the vehicle to carry out movements and stabilize the vehicle.

For the BROV2 Heavy equipped with eight thrusters, this can be expressed quantitatively,

$$\boldsymbol{\tau} = \boldsymbol{T}\boldsymbol{m} \tag{3.1}$$

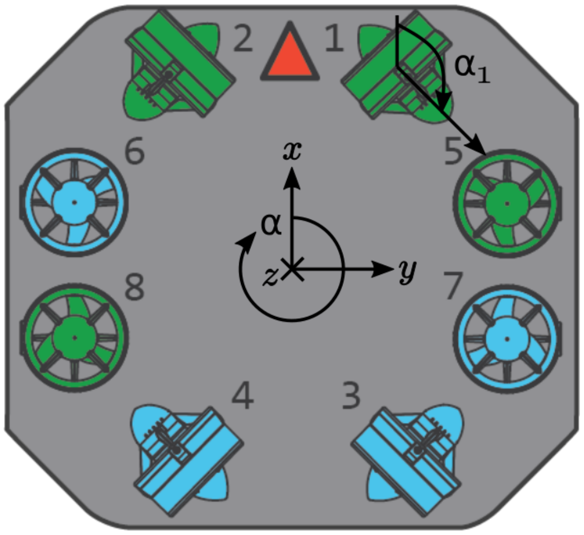


Figure 3.1: A diagram of the eight BROV2 Heavy thrusters with coordinate axes and indices labeled. The BROV2 Heavy has four horizontal azimuthal thrusters and four vertical upright thrusters; the red triangle indicates the forward orientation of the vehicle; propellers of the thrusters shown with the color green (1, 2, 5 and 8) rotate counter-clockwise and those in blue (3, 4, 6, and 7) rotate clockwise, i.e., for the vertical thruster, the green ones have negative rotation about the z-axis and the green ones have positive rotation about the z-axis; the force resultant on the vehicle from positive forward thrust is in the direction of the more textured half of the azimuthal thrusters and downwards (into the page) for the vertical thrusters [37].

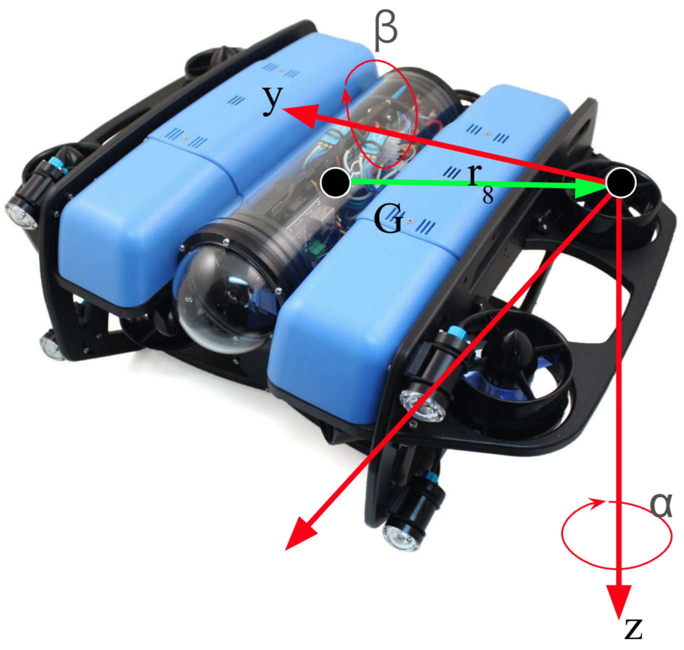


Figure 3.2: An annotated photograph of the BROV2 showing the vector locating the position of the eighth thruster $\boldsymbol{r_8}$ with respect to the center of gravity G. Angular orientation is designated with α and β for rotations about the local y-axis and z-axis, respectively.

Table 3.1: Locations in the body-fixed frame of the eight T200 thrusters on the BROV2 Heavy [38].

$\boldsymbol{r_i}$	$\boldsymbol{r_i} \cdot \hat{\boldsymbol{i}}$ (m)	$\boldsymbol{r_i} \cdot \hat{\boldsymbol{j}}$ (m)	$\boldsymbol{r_i} \cdot \hat{\boldsymbol{k}}$ (m)
$\boldsymbol{r_1}$	0.156	-0.111	0.085
$\boldsymbol{r_2}$	0.156	-0.111	0.085
$\boldsymbol{r_3}$	-0.156	-0.111	0.085
$\boldsymbol{r_4}$	-0.156	-0.111	0.085
$\boldsymbol{r_5}$	0.120	-0.218	0
$\boldsymbol{r_6}$	0.120	0.218	0
$\boldsymbol{r_7}$	-0.120	0.218	0
$\boldsymbol{r_8}$	-0.120	-0.218	0

where $\boldsymbol{\tau}$ is the six-by-one vector of control forces and torques, $\boldsymbol{T}$ is the six-by-eight thruster allocation matrix [35], and $\boldsymbol{m}$ is the eight-by-one vector of the force contributions of the eight thrusters. The constant matrix $\boldsymbol{T}$ is a linear transformation that maps the force and torque inputs of the motors $\boldsymbol{m}$ to the overall resultant forces and torques on the vehicle body $\boldsymbol{\tau}$. This is the same $\boldsymbol{\tau}$ that is determined by the control law in Equation (5.20).

The first three rows of $\boldsymbol{T}$ transform the eight thruster input forces into the resultant body linear forces and are constructed column-wise with

$$\boldsymbol{T}_{1:3,i} = [\cos(\alpha_i)\ \sin(\alpha_i)\ \sin(\beta_i)]^\top, \tag{3.2}$$

where α_i is the angle of the i-th thruster in the horizontal plane of the vehicle and β_i is the angle of the i-th thruster in roll (see Figure 3.1). All the force unit vectors for the eight thrusters' orientations put together form the top three rows of $\boldsymbol{T}$; which when given an input $\boldsymbol{m}$, outputs the linear forces (i.e., the first three entries of $\boldsymbol{\tau}$). Segments of vectors will be referenced with indexed ranges "$a : b$," where a is the starting index and b is the ending index. Figure 3.1 displays the FRD body frame coordinate axes, relative locations, and orientations of the thruster configuration. Thruster 1 is annotated as an example of how α_1 is defined: the angle α follows the right-hand convention for clockwise yaw rotation about the downwards-pointing z-axis; β is not visible in the figure but also follows the right-hand convention for pitch rotation about the y-axis.

The last three rows of $\boldsymbol{T}$ map the thruster force inputs to the resultant body torques and are constructed column-wise with

$$\boldsymbol{T}_{4:6,i} = \boldsymbol{r}_i \times \boldsymbol{T}_{1:3,i}. \tag{3.3}$$

where $\boldsymbol{r_i}$ is the location of the i-th thruster relative to the vehicle's center of mass, and a cross product is performed with it and the first three rows of the i-th column of $\boldsymbol{T}$ determined earlier in Equation (3.2). Similar to how Equation (3.2) is a unit vector for forces, Equation (3.3) is a unit vector for torques. All the torque unit vectors for the eight thrusters are put together to form the bottom three rows of $\boldsymbol{T}$, which when given an input $\boldsymbol{m}$, output the angular torques (last three entries of $\boldsymbol{\tau}$).

Vehicle system geometry presented in [38] is reproduced in Table 3.1. By concatenating the eight vectors resulting from Equations (3.2) and (3.3), the total thruster allocation matrix for the BROV2 Heavy is

$$\boldsymbol{T} = \begin{bmatrix} -0.707 & -0.707 & 0.707 & 0.707 & 0 & 0 & 0 & 0 \\ 0.707 & -0.707 & 0.707 & -0.707 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 & 1.0 & 1.0 & 1.0 \\ -0.060 & 0.060 & -0.060 & 0.060 & 0.218 & -0.218 & 0.218 & -0.218 \\ -0.060 & -0.060 & 0.060 & 0.060 & -0.120 & -0.120 & 0.120 & 0.120 \\ 0.189 & -0.189 & -0.189 & 0.189 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.4}$$

shown here rounded to three decimal places. Given a vector $\boldsymbol{m}$ containing the variable thrust of all eight thrusters, Equations (3.1) and (3.4) is used to determine the control forces and torques $\boldsymbol{\tau}$ resulting from the thrusters on the BROV2 body in the updated simulation model.

Now, consider if some desired forces and torques on the BROV2 were given and the necessary motor activations $\boldsymbol{m}$ needed to be calculated instead, as is the case of an autopilot mechanism; the inverse thruster allocation matrix would be required. That is present in

$$\boldsymbol{m} = \boldsymbol{T}^{\dagger}\boldsymbol{\tau} \tag{3.5}$$

where $\boldsymbol{T}^{\dagger}$ is the pseudo-inverse of the rectangular matrix $\boldsymbol{T}$. The pseduo-inverse is used because the thruster allocation matrix is a rectangular matrix that which does not have a true inverse [39]. The specific matrix computed from the pseudo-inverse of Equation (3.4) has values of

$$\boldsymbol{T}^{\dagger} = \begin{bmatrix} -0.354 & 0.354 & 0 & 0 & 0 & 1.324 \\ -0.354 & -0.354 & 0 & 0 & 0 & -1.324 \\ 0.354 & 0.354 & 0 & 0 & 0 & -1.324 \\ 0.354 & -0.354 & 0 & 0 & 0 & 1.324 \\ 0.177 & 0.097 & 0.250 & 1.147 & -2.083 & 0 \\ 0.177 & -0.097 & 0.250 & -1.147 & -2.083 & 0 \\ -0.177 & 0.097 & 0.250 & 1.147 & 2.083 & 0 \\ -0.177 & -0.097 & 0.250 & -1.147 & 2.083 & 0 \end{bmatrix}. \tag{3.6}$$

The matrix above is used to transform an input vector of desired forces and torques on the vehicle body $\boldsymbol{\tau}$ into the required motor activations $\boldsymbol{m}$ to achieve them. These arrays $\boldsymbol{T}$, $\boldsymbol{m}$, $\boldsymbol{\tau}$, and $\boldsymbol{T}^{\dagger}$ have compatible physical units in the Meters–Kilogram–Second (MKS) system of measurements. For more details of the programming process to implementation of $\boldsymbol{T}$ to the BROV2 firmware and $\boldsymbol{T}^{\dagger}$ into the ArduSub simulation see Appendix B.

3.2 Thruster Model

When the firmware is processing an input of desired force/torque, before spinning up the thrusters it calculates the thrust vector $\boldsymbol{m}$ for the eight motors with the inverse thruster allocation matrix as in Equation (3.5). Each thruster must spin at velocities dependent on the value of $\boldsymbol{m}$. The BROV2 activates its thrusters using electronic speed controllers (ESCs) that are controlled with pulse-width modulation (PWM) signals. In order to determine what PWMs to send to the ESCs, the firmware was updated to reference a lookup table based on the experimental data collected by Blue Robotics [40] and graphed in Figure 3.3.

There are a total of eighty-one tested PWMs, let $j \in \{1, 2, ..., 81\}$ be an index representing the thrust–PWM pairs (F_j, P_j). For any given m_i, $i \in 1, 2, ...8$ the lookup table function finds the closest two thrusts such that $F_j < m_i < F_{j+1}$, where j is the index of the closest thrust below the commanded thrust. Then the lookup table function uses linear interpolation to solve for what PWM

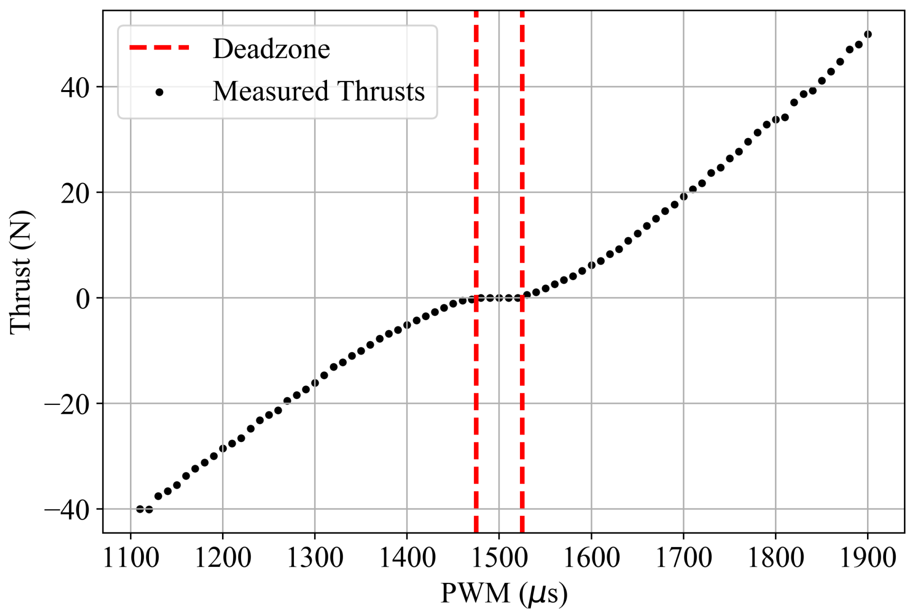


Figure 3.3: Blue Robotics experimental test results of pulse width modulation (PWM) input and resulting thrust for a single T200 thruster.

to send to the ESC. In equation form that linear interpolation is

$$P_i = \begin{cases} P_1, & \text{if } m_i \leq F_1, \\ \text{round}\left(\dfrac{m_i - F_j}{F_{j+1} - F_j}(P_{j+1} - P_j) + P_j\right), & \text{if } F_j < m_i < F_{j+1}, \\ P_{81}, & \text{if } m_i \geq F_{81}. \end{cases} \tag{3.7}$$

where P_i is the calculated PWM sent to the i-th ESC, round$(\cdot)$ is a function to round the value to the nearest integer, m_i is the commanded thrust for the i-th motor, F_j is the closest experimental thrust below m_i, F_{j+1} is the closest experimental thrust above m_i, P_j is the corresponding experimental PWM to F_j, and P_{j+1} is the corresponding experimental PWM to F_{j+1}. In the case that m_i is outside the range of the experimentally determined thrusts, the minimum or maximum PWM is output. The lookup table was implemented on the firmware to have an accurate model of the T200 when allocating thrusts given desired forces and torques. The lookup table was also implemented in the simulation environment to accurately simulate the output thrusts effect on a virtual BROV2. For more details about firmware and software implementation, see Appendix B.

Chapter 4

Characterizing Hydrodynamics with Open-Loop Testing

In this chapter the preliminary testing that was performed before calibrating the control law shall be discussed. This preliminary testing was necessary because it was found that existing values found in the literature for the hydrodynamic coefficients following the model proposed in Chapter 2 were invalid for the BROV2 Heavy in-house. The experimental methods described in this chapter produced hydrodynamic coefficients that modeled the BROV2 Heavy fairly well and serve as a base for any future changes. If the vehicle were to be equipped with any auxiliary equipment like new sensors or a manipulator arm, the hydrodynamic model would change and could be reassessed by performing the same experiments described here.

Hydrodynamic coefficients produced by the analysis in this chapter updated the "simulation model" and the "linear approximation model." The simulation model is a nonlinear model and directly uses the coefficients; whereas how the coefficients are used to produce the linear approximation model using trim conditions shall be covered in detail in Chapter 5. ArduSub has a feature known as simulation-in-the-loop (SITL), which allows users to test their customized firmware or programs virtually. This is potentially a useful tool, because changes to the firmware can be tested directly because the binary compilation process of the virtual SITL firmware is the same as the actual firmware and programs written for the vehicle can be tested verbatim without needing to physically deploy the vehicle. However, the existing SITL uses a very crude approximation of the BROV2 Heavy, making it useful for simple tests like verifying inputs and outputs. Updates to the simulation

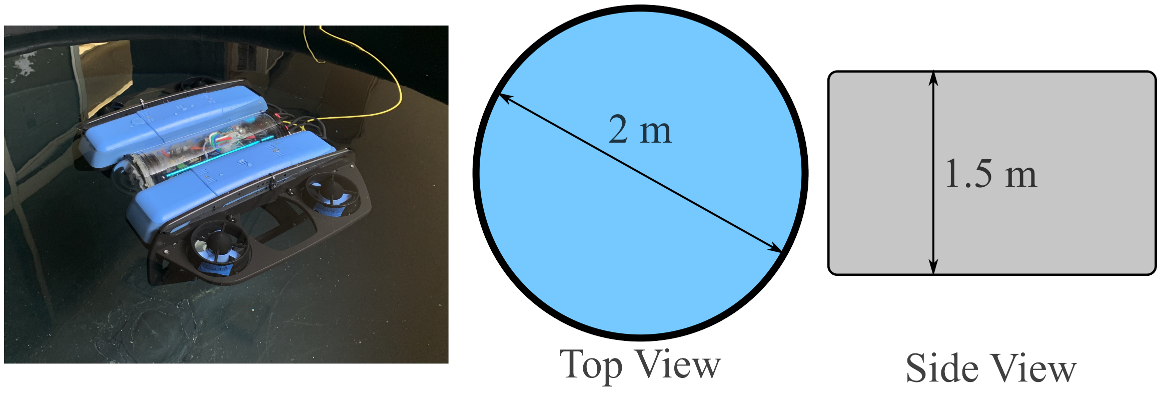


Figure 4.1: A photograph of the BROV2 floating in the test tank and illustrations of the test tank with approximate dimensions.

model, make SITL a more valuable tool that can produce results representative of performance in the field.

As a first step, hydrodynamic coefficients in the heave and yaw directions are characterized, as these two are the most crucial degrees of freedom in terms of station keeping, and are meant to serve as characteristic representations of both linear and angular degrees of freedom. The untested degrees of freedom were scaled proportionally to the experimentally determined hydrodynamic coefficients with the ratios of values from the literature. All final values of the hydrodynamic coefficients are tabulated at the end of this chapter.

4.1 Yaw Degree of Freedom

In this section, a description of the open-loop yaw torque experimental testing that was performed to determine the yaw hydrodynamic coefficients for hybrid autopilot model and ArduSub SITL shall be covered. The yaw degree of freedom isolated from the full governing dynamics model presented in Equation 2.14 is

$$N = (I_{zz} + N_{\dot{r}})\dot{r} + N_r r + N_{r|r|} r^2 \tag{4.1}$$

where N is the external yaw torque acting on the BROV2, I_{zz} is the moment of inertia in yaw, $N_{\dot{r}}$ is the added mass in yaw, N_r is the linear viscous damping in yaw, $N_{r|r|}$ is the quadratic damping in yaw, r is the vehicle's yaw velocity, and $\dot{r}$ is the vehicle's yaw acceleration. (Note how centripetal/Coriolis terms are absent in this equation because motion in all other degrees of freedom ar assumed to be zero.)

Preliminary results of real-world testing of the yaw step response using critically damped gains based on BROV2 models existing in the literature produced large discrepancies between the settling times predicted by SITL, where actual settling times were significantly longer. Therefore, water tank testing for system identification was performed that determined a more accurate model for simulations and the hybrid autopilot.

Open-loop testing of the yaw degree of freedom of the BROV2 Heavy was performed in a water tank (Figure 4.1). During this testing, six different yaw torques were applied to the BROV2 and the resulting yaw velocities were measured, with at least five trials performed per torque tested. These tests were intended to observe unsteady behavior that can be used for fitting drag and added mass terms. The particular BROV2 Heavy used in these experiments had additional ballast in the form of metal washers attached to the front corners of the vehicle frame to level out its resting floating state. It was slightly positively buoyant, and before each trial was allowed to come to rest near the surface of the water, mostly submerged so that less than a quarter inch of the buoyancy floats was exposed. An experimenter also held the vehicle's tether above and out of the way, careful not to disturb the vehicle.

At the start of each trial at time $t = 0$, a fixed yaw torque input command was sent to the vehicle and held constant. Data collection recorded yaw velocity r for times $t \geq 0$. Yaw torques were held constant for several seconds to give the velocity output ample time to stabilize before ceasing the trial. Each open-loop response had two distinct regions of the velocity output: an initial unsteady region while the vehicle was accelerating from rest and a steady-state region where the vehicle was no longer accelerating and maintained a constant velocity r_f. The unsteady region of the velocity output of the open-loop response was analyzed to determine inertia characteristics and the steady-state region to determine drag characteristics.

4.1.1 Viscous Drag Analysis

Given a constant yaw torque, the BROV2 starting from rest eventually reaches an equilibrium state after enough time has elapsed, where the drag torque balances with the thruster torque. While in equilibrium, acceleration largely ceases and a steady-state velocity is maintained. Equation (4.1) is reduced to

$$N(\dot{r} = 0) = N_r r + N_{r|r|} r^2. \tag{4.2}$$

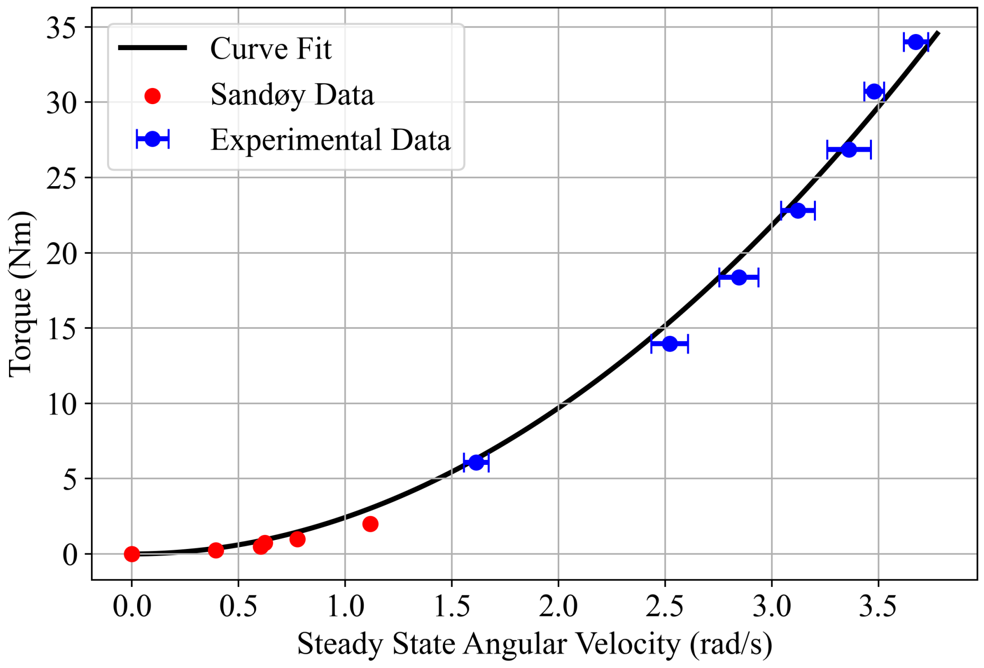


Figure 4.2: Experimental results from the drag analysis of the open-loop yaw testing shown in blue; the complete drag function is actually symmetric about the origin. Error bars were calculated as two sample standard deviations to the left and right of the mean. Results are found to be comparable to prior research on a similar vehicle shown in red [41].

Previous work in [41] tested a smaller ROV similar to the BROV2 at low torques; in this work, a variety of test torques above this range were tested.

It was found that all open-loop responses at the various torques reached a steady state within two and a half seconds after the input torque was initiated. Because the experimental velocity data collected were noisy, a two-second interval spanning the range $2.5\,\mathrm{s} \leq t \leq 4.5\,\mathrm{s}$ was averaged to obtain the steady-state velocity. Mean and variance measurements were calculated for each yaw torque value tested and are recorded in Figure 4.2. Using the experimental results, a least squares curve was fit to Equation (4.2) and plotted with the open-loop experimental data as well as the data from [41]. It can be seen that there is good agreement between the fitted curve across the entire range of experimental torques and velocities. Even though the curve was fit to the open-loop yaw experiments of this work, the curve also aligns fairly well to data from [41]. The fitted hydrodynamic coefficients for this curve found that the linear term N_r was zero and the resulting quadratic term value for $N_{r|r|}$ is recorded in Table 4.1.

4.1.2 Inertia Analysis

Every BROV2 has unique inertia properties resulting from their payload, which may consist of custom modules or different sensors. In the case of the BROV2 that underwent open-loop testing, the ballasts stabilized its buoyancy altered its angular moments of inertia. Having determined that there is no linear term N_r, and constraining the value of $N_{r|r|}$ according to the analysis just described, the acceleration term $\dot{r}$ (Equation 4.1) may be reintroduced to study the unsteady regime and determine the remaining inertia terms. For this analysis added mass $N_{\dot{r}}$ assumes the values suggested in [38] which estimated the BROV2 added mass based on a similarly shaped, albeit smaller, vehicle [41]. The added mass for the similar vehicle was calculated using an analytical method that took added mass properties known for simple geometric shapes and combined them with scaling factors to approximate the ROV [42]. The input torque N controlled by the experimenter is also considered constrained, thus the only remaining unknown quantity in Equation (4.1) is the moment of inertia I_{zz}.

Stated mathematically, Equation (4.1) reduces to the ordinary differential equation

$$\dot{r}(N) = \frac{N - N_{r|r|}r^2}{I_z + N_{\dot{r}}} \tag{4.3}$$

where $\dot{r}(N)$ is the vehicle's yaw acceleration given an applied input yaw torque N. Hence, for any of the six yaw torques tested Equation (4.3) has a one-parameter family of solutions

$$r(N) = f(t, I_z) \tag{4.4}$$

where r is yaw velocity, which is a function of time t, respective control torque N, and moment of inertia I_{zz}.

The solutions proposed above are applicable for constant torques N. Referencing Figure 4.3, it can be seen that in the moments immediately after the yaw torque input initiates there is a concave up region where the yaw torque is ramping up and has not yet reached its intended magnitude. Every trial has this initial thruster ramp-up time when yaw torque increases with time; the velocity at the time when constant torque is reached describes the left-hand boundary conditions for the added mass fitting. While a constant torque is maintained, velocity continues to increase with time. Eventually, in every trial the vehicle reaches an equilibrium steady state as defined in the previous

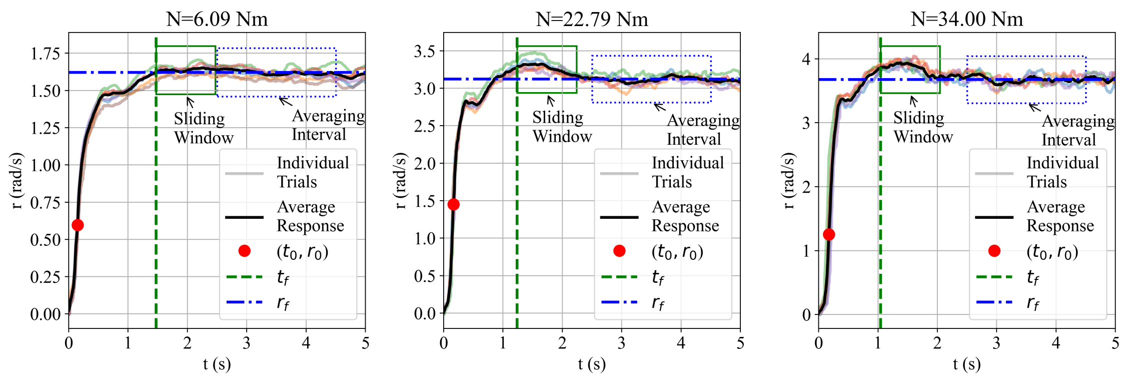


Figure 4.3: A few examples of the open-loop yaw response analysis at different commanded yaw torques. The yaw heading data contained in the dotted blue box is the averaging interval to determine steady-state yaw velocity r_f first, shown with the dash-dotted horizontal blue line. Then a sliding window moving average shown with a solid green box was used to determine settling time t_f, shown by the dashed vertical green line.

viscous drag analysis. Therefore, the intervals and boundary conditions of Equation (4.4) are

$$t_0 \leq t \leq t_f, \quad r(t_0) = r_0, \quad r(t_f) = r_f. \tag{4.5}$$

where t is time, t_0 is the thruster ramp-up time while the vehicle is accelerating, t_f is the settling time when steady-state velocity is reached, r_0 is the velocity when constant torque is reached, and r_f is the steady-state velocity. At the time t_0, where the curve transitions from concave up to concave down, full respective constant torque is reached; the corresponding velocity at the inflection point also gives the initial conditions $r(t_0) = r_0$ for the particular solution. Extracting t_f from the experimental data required other processing. To account for noise and the overshoot present at higher torques, a sliding window that calculated a moving average starting from the initial overshoot moving to the right was utilized, and the settling time was determined to be reached when the sliding window average absolute error from r_f was within a 5% tolerance, choosing the left-hand limit of the sliding window as t_f.

Now that the differential Equation (4.3) is well defined, with all other constants constrained, and the boundary conditions are established, the ordinary differential Equation (4.4) poses initial value problems for $r(N)$ dependent on the choice of I_{zz}.

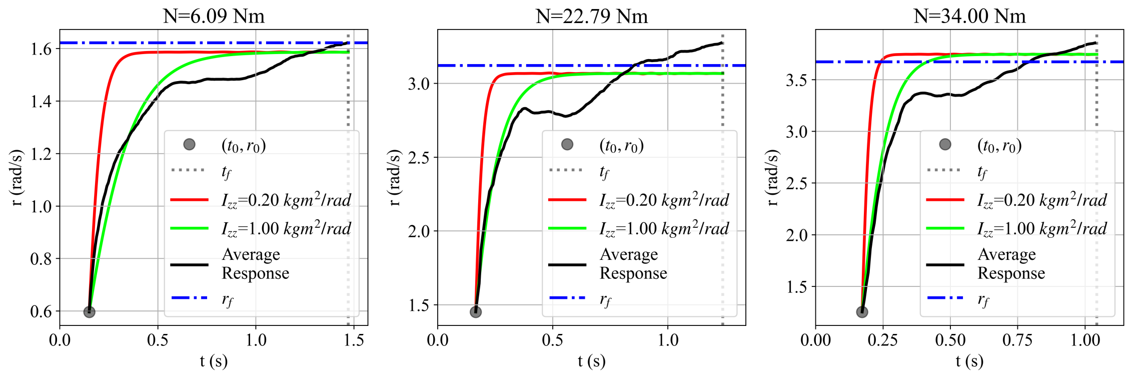


Figure 4.4: Comparison of the solutions of the lower moment of inertia in the yaw direction I_{zz} suggested by [38, 43, 41] alongside the chosen $I_{zz} = 1.0\,\mathrm{kg\,m^2/rad}$.

Essentially, Equation (4.3) is an initial value problem with the boundary conditions shown in Equation (4.5) having a one-parameter family of solutions Equation (4.4) dependent on I_{zz}. To trim down the search space, I_{zz} was assumed to be greater than the values suggested in [43, 38, 41] and less than a solid rectangle with the dimensions of the BROV2. A set of values within this range was iteratively tested against the actual unsteady response recorded in the experimental data using the root mean square error along the $t_0 \leq t \leq t_f$ interval. Results of this analysis are shown in Figure 4.4. The best agreement was found at $I_{zz} = 1.0\,\mathrm{kg\,m^2/rad}$.

While I_{zz} was treated as the unknown quantity that was solved with the ODE analysis, what was actually determined was a bulk inertia term encompassing both the moment of inertia and added mass in yaw. Because I_{zz} and $N_{\dot{r}}$ both linearly scale with $\dot{r}$, the precise distribution is unknown. However, the exact breakdown of each term is not important since all modeling/simulation utilizes the combined inertia term.

4.2 Heave Degree Of Freedom

In this section the experimental method for determining the heave motion hydrodynamic model shall be discussed. Section 2.2.3 explains that there is a significant asymmetry across the x_b-y_b plane of the BROV2 causing the hydrodynamic coefficients for heave to be change dependent on the direction of heave motion. Equations (2.27) and (2.28) display the model for heave isolated from the full governing dynamics model (Equation (2.14)). Equations (2.27) and (2.28) reproduced here for convenience,

$$Z = (m + Z_{\dot{w}})\dot{w} + Z_w w + Z_{w|w|} w^2 + \boldsymbol{F_B} \cdot \hat{\boldsymbol{k}} \tag{4.6}$$

where Z is the body heave force, m is the mass of the BROV2 Heavy, $Z_w \dot{w}$ is the added mass in heave, Z_w is the linear viscous damping in heave, $Z_{w|w|}$ is the quadratic damping in heave, w is the vehicle's heave velocity, $\dot{w}$ is the vehicle's heave acceleration, and $\boldsymbol{F_B} \cdot \hat{\boldsymbol{k}}$ is the vertical component of the body-frame restoring forces vector (Equations 2.23). The coefficients are defined by the piecewise functions

$$Z_{\dot{w}} = \begin{cases} Z_{\dot{w},rise} & \text{if } \dot{w} < 0, \\ Z_{\dot{w},dive} & \text{if } \dot{w} > 0, \end{cases} \quad Z_w = \begin{cases} Z_{w,rise} & \text{if } w < 0, \\ Z_{w,dive} & \text{if } w > 0, \end{cases} \quad Z_{w|w|} = \begin{cases} Z_{w|w|,rise} & \text{if } w < 0, \\ Z_{w|w|,dive} & \text{if } w > 0. \end{cases} \tag{4.7}$$

Here, heave acceleration and velocity are in the FSD frame.

Experimental data for the BROV2 Heavy in the positive and negative heave directions was gathered in anticipation of a piecewise model of the form above. Using the same water tank as the yaw open-loop testing of the previous section the heave open-loop testing was performed where the vehicle was moved through the approximately one and half meter vertical water distance. A very tall tank would have been best in order for steady-state velocity to be observed. Unfortunately the tank was not tall relative to the height of the BROV2 and its maximum possible control heave forces.

It was not known ahead of time when velocities would become dangerous so tests were performed starting from very low heave forces that barely overcame the buoyancy forces and increased in gradual increments. Heave motion data was collected for about twenty-five percent of the maximum control heave, equivalent to thirty-eight tests. Tests consisted of a constant dive force command followed by a constant rise force command, and were conducted three times each dive/rise force pair. Each test collected data for the angular orientation, heave acceleration, depth, and commanded force.

4.2.1 Preparing Open-Loop Data

First, all data was interpolated to share the same discrete times. The exact condition desired to be observed is when a constant heave force is moving the completely submerged vehicle through the water, to align with the theoretical heave hydrodynamic model Equation (4.6). Data was organized and filtered with cropping criteria to best isolate this condition and match the theoretical model.

Each command heave was also expected to have a spin-up time during which the thrusters are ramping up from rest to the commanded heave. In addition to that, every dive sequence was expected to have some regime during which the BROV2 was overcoming the surface effects of the water; similarly, during rise sequences a regime was observed where an unexpected sign change of acceleration occurred near the surface of the water.

To reject unmodeled phenomena like thruster ramp up and surface effects cropping criteria was chosen based on the times when the BROV2 crossed certain depths. Every dive trial data was trimmed to the times during which the BROV2 dove between the depths $-0.3\,\mathrm{m} \leq z - 0.7\,\mathrm{m}$. Every rise trial data was trimmed to the times after which the vehicle rose past $z \geq -0.7\,\mathrm{m}$ and stopped when the sign of the FSD heave acceleration $\dot{w}$ changed from negative to positive. After applying cropping criteria each of the three trials per tested dive and heave force were trimmed at the tail end of their durations to the shortest of the trials being averaged together. Finally each three trial group of dive and rise data per test were averaged separately.

4.2.2 Least Squares Analysis

Let the design matrix be represented with $\boldsymbol{X}$. Let the columns be pertinent data vectors of independent variables stacked in ascending order of i-th test number, resulting in

$$\boldsymbol{X_{i1}} = \begin{bmatrix} \dot{\boldsymbol{w}}_{\boldsymbol{1}} \\ \dot{\boldsymbol{w}}_{\boldsymbol{2}} \\ \vdots \\ \dot{\boldsymbol{w}}_{\boldsymbol{i}} \\ \vdots \\ \dot{\boldsymbol{w}}_{\boldsymbol{38}} \end{bmatrix}, \quad \boldsymbol{X_{i2}} = \begin{bmatrix} \boldsymbol{w_1} \\ \boldsymbol{w_2} \\ \vdots \\ \boldsymbol{w_i} \\ \vdots \\ \boldsymbol{w_{38}} \end{bmatrix}, \quad \boldsymbol{X_{i3}} = \begin{bmatrix} \boldsymbol{w_1} \odot |\boldsymbol{w_1}| \\ \boldsymbol{w_2} \odot |\boldsymbol{w_2}| \\ \vdots \\ \boldsymbol{w_i} \odot |\boldsymbol{w_i}| \\ \vdots \\ \boldsymbol{w_{38}} \odot |\boldsymbol{w_{38}}| \end{bmatrix}, \tag{4.8}$$

such that $\odot$ represents the element-wise multiplication of two vectors producing another vector and the terms $\boldsymbol{w_i} \odot |\boldsymbol{w_i}|$ are the signed squared velocities of every discrete element of the i-th test. Notice that each row of X_{ij} represents a time instance of all data for a given test.

Let the three-by-one vector of unknown scalar parameters be $\boldsymbol{\beta}$, such that

$$\boldsymbol{\beta} = \begin{bmatrix} m + Z_{\dot{w}} \\ Z_w \\ Z_{w|w|} \end{bmatrix}. \tag{4.9}$$

Let the dependent variable be the net body force in heave, accounting for buoyancy

$$\boldsymbol{y_i} = \begin{bmatrix} \boldsymbol{Z_1} - \boldsymbol{F}_{\boldsymbol{1,B}}^{\top}\hat{\boldsymbol{k}} \\ \boldsymbol{Z_2} - \boldsymbol{F}_{\boldsymbol{2,B}}^{\top}\hat{\boldsymbol{k}} \\ \vdots \\ \boldsymbol{Z_i} - \boldsymbol{F}_{\boldsymbol{i,B}}^{\top}\hat{\boldsymbol{k}} \\ \vdots \\ \boldsymbol{Z_{38}} - \boldsymbol{F}_{\boldsymbol{38,B}}^{\top}\hat{\boldsymbol{k}} \end{bmatrix}. \tag{4.10}$$

The equation

$$\boldsymbol{y} \approx \boldsymbol{X}\boldsymbol{\beta} \tag{4.11}$$

states that the theoretical net heave force $\boldsymbol{y}$ should be approximately equal to the vectorized form of the experimental data $\boldsymbol{X}$ multiplied by some unknown coefficients $\boldsymbol{\beta}$, matching the form of the heave hydrodynamic model (2.27). The product of the design matrix and parameters $\boldsymbol{X}\boldsymbol{\beta}$ is the projected heave. Depending on the choice of the parameters there is some error between theoretical and projected values $\boldsymbol{\epsilon}$. The least squares method aims to minimize the norm of this error by determining optimal parameters. Stated mathematically,

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{\epsilon}\|^2 = \min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|, \tag{4.12}$$

where $\|\cdot\|$ is the norm of a vector. The exact method of least squares to determine $\boldsymbol{\beta}$ involves numerical matrix manipulation methods [39, 44] that were implemented with scientific computer libraries.

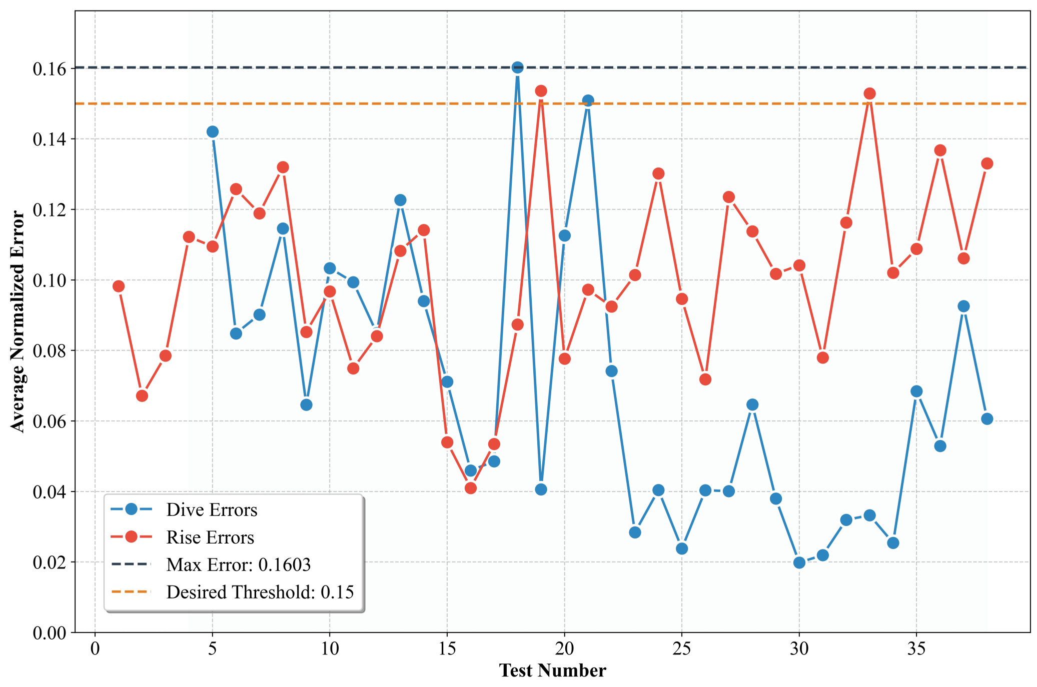


Figure 4.5: Average normalized error for the theoretical dive and rise heave models compared to heave forces predicted by hydrodynamic coefficients determined by least squares fitting of experimental data.

One advantage to extensively testing low heave forces is that linear drag Z_w, expressed most at low velocities can be studied well. The other side of that coin is that there is less confidence in the sufficiency of data for the quadratic drag $Z_{w|w|}$ fitting, which is most important for the high forces required for responsive control.

Figure 4.5 shows the average normalized error defined as

$$\overline{e_i} = \frac{1}{T_i} \int_0^{T_i} \frac{|\boldsymbol{X_{ij}} \beta_j - \boldsymbol{y_i}|}{\boldsymbol{y_i}} dt, \tag{4.13}$$

where i is the index number of the test set, T_i is the length of time of the average dive or rise. The first four dive tests were removed from the inputs to the least squares fitting because their errors greatly exceeded the desired fifteen percent error threshold. Except for a few exceptions most tests fell below the error threshold, the chosen hydrodynamic coefficients for (4.7) are recorded in Table 4.1

4.3 Hydrodynamic Coefficients

Values for the hydrodynamic coefficients are gathered in Table 4.1. Mass was found in the specifications sheet for the BROV2 Heavy provided by Blue Robotics [23]. The net buoyancy parameter (Section 2.2.2) was determined by incrementally adding weights to the vehicle while it floated on the surface, the amount of additional ballast that caused the vehicle to sink very slowly was recorded. Yaw inertia, linear damping, and quadratic damping values were determined with the analysis of the open-loop testing described in Section 4.1. Pitch hydrodynamic coefficients were set equal to yaw, whereas the roll hydrodynamic coefficients were scaled to around 86% of the former to match the proportionality suggested by the ratios of rotational inertia terms in [43]. Added mass, linear damping, and quadratic damping in heave followed a piecewise model determined with experimental testing and analysis as described in Section 4.2. The proportionality of models for the linear degrees of freedom suggested in [38], based on added mass estimations [31, 42] and tow tank testing [41], were used to determine the surge and sway hydrodynamics with respect to the average of the heave (rise) and the heave (dive) coefficients for each respective parameter type.

Another notable value important to the quantified hydrodynamic model is the location of the center of buoyancy

$$\boldsymbol{r_b} = [0, 0, -0.02\,\mathrm{m}]^\top \tag{4.14}$$

this is the body-fixed frame vector pointing from the center of gravity to the center of buoyancy discussed in Section 2.2.2.

Mass and Inertia Parameters			
Parameter	Symbol	Units	Value
Mass	m	kg	11.5
Net Buoyancy	$\boldsymbol{F_{B,net}} \cdot \boldsymbol{\hat{k}}$	N	6.675
Roll Inertia	I_{xx}	kg·m^2	0.8571
Pitch Inertia	I_{yy}	kg·m^2	1.0
Yaw Inertia	I_{zz}	kg·m^2	1.0

Linear Coefficients (Added Mass)			
DoF	Symbol	Units	Value
Surge	$X_{\dot{u}}$	kg	10.77
Sway	$Y_{\dot{v}}$	kg	24.86
Heave (Dive)	$Z_{\dot{w},dive}$	kg	34.60
Heave (Rise)	$Z_{\dot{w},rise}$	kg	22.45
Roll	$K_{\dot{p}}$	kg·m^2/rad	0.103
Pitch	$M_{\dot{q}}$	kg·m^2/rad	0.120
Yaw	$N_{\dot{r}}$	kg·m^2/rad	0.120

Linear Damping Coefficients			
DoF	Symbol	Units	Value
Surge	X_u	Ns/m	38.95
Sway	Y_v	Ns/m	60.11
Heave (Dive)	$Z_{w,dive}$	Ns/m	49.50
Heave (Rise)	$Z_{w,rise}$	Ns/m	50.62
Roll	K_p	Ns/rad	0.0
Pitch	M_q	Ns/rad	0.0
Yaw	N_r	Ns/rad	0.0

Quadratic Damping Coefficients			
DoF	Symbol	Units	Value
Surge	$X_{u\|u\|}$	Ns^2/m^2	31.01
Sway	$Y_{v\|v\|}$	Ns^2/m^2	36.95
Heave (Dive)	$Z_{w\|w\|,dive}$	Ns^2/m^2	113.86
Heave (Rise)	$Z_{w\|w\|,rise}$	Ns^2/m^2	76.69
Roll	$K_{p\|p\|}$	Ns^2/rad^2	2.08
Pitch	$M_{q\|q\|}$	Ns^2/rad^2	2.42
Yaw	$N_{r\|r\|}$	Ns^2/rad^2	2.42

Table 4.1: A collection of tables showing BlueROV2 Heavy hydrodynamic parameters.

Chapter 5

Controller Design

This chapter shall lead with a derivation of the foundational control law. The difficulties encountered when actually calibrating the control laws for the individual degrees of freedom shall be recounted. First the yaw degree of freedom will be reviewed and the chapter will finish with the heave degree of freedom.

5.1 Control Law Based on Second Order Linear Time-Invariant Approximation

In this section the Linear Time-Invariant (LTI) approximation for the BROV2 hydrodynamic model is proposed. A simple derivation of a control law for a single degree of freedom is performed. Then the derivation of finding the critically damped gains for the 6DOF model using matrix algebra is performed.

5.1.1 Control Law for a Single Degree of Freedom

Linear Time-Invariant (LTI) systems are dynamic systems which are understood very well and have used Proportional-Derivative (PD) error feedback controllers to great effect. The key difference between the hydrodynamic model explained in the previous section and an LTI model is the drag (2.18) that scales quadratically with velocity producing nonlinearities. Despite this difference, the hydrodynamic model can be approximated as an LTI system to serve as a template for the PD controller model.

The controller model is built one degree of freedom at a time as a simplification so that the response of the system to the control input can be solved analytically, and therefore control parameters can be derived to produce desirable behavior. For explanation purposes let γ be a state variable representing a single degree of freedom. Proportional error is defined as the difference between desired state and actual state,

$$\tilde{\gamma} = \gamma_d - \gamma \tag{5.1}$$

where $\tilde{\gamma}$ is proportional error, γ_d is desired state, and γ is the actual state. Derivative error is defined as the time rate difference between desired velocity and actual velocity,

$$\dot{\tilde{\gamma}} = \frac{\dot{\tilde{\gamma}}}{dt} = \dot{\gamma}_d - \dot{\gamma} \tag{5.2}$$

where $\tilde{\gamma}$ is the derivative error, $\dot{\gamma}_d$ and $\dot{\gamma}$ are the time rate of change of desired state and actual state, respectively.

A control law is defined such that the force input to the system from the thrusters, at any point in time, is calculated as a function of the values of $\tilde{\gamma}$ and $\dot{\tilde{\gamma}}$ driving the system towards its desired state autonomously. Scalars known as control gains weight how much effect the proportional and derivative errors have in the control law

$$\tau = k_P\tilde{\gamma} + k_D\dot{\tilde{\gamma}} \tag{5.3}$$

where τ is the force/torque commanded by the controller, k_P is the proportional gain, and k_D is the derivative gain.

Linear control laws are often spoken in terms of their step responses. In summary, the proportional term $k_P\tilde{\gamma}$ generates force or torque that controls how quickly the system is driven towards the desired state. The derivative term $k_D\dot{\tilde{\gamma}}$ is also known as the damping term because it slows down the response to prevent overshoot and restrict oscillations.

5.1.2 Calculating Critically Damped Gains for Full 6DOF Hydrodynamic Model

The control forces/torques in $\boldsymbol{\tau}$ of Equation (5.3) act on the system as described with the full physical dynamic model from Equation 2.14. Using techniques from linear control theory, gains for a critically damped second order system as a function of desired closed loop natural frequency can be calculated in terms of the dynamic model parameters. In terms of linear control theory, that describe systems in terms of step response behavior, a critically damped system is one that reaches a desired state as fast as possible without overshoot. Too great proportional gain or too little derivative gain results in underdamped oscillations, and the inverse causes a slow overdamped response. When considering an AR/VR pilot controlling the BROV2 with immersive equipment, a critically damped system would be ideal to track gestures quickly while also preventing motion sickness. Thus, a new matrix replaces $\boldsymbol{B_L}$ and $\boldsymbol{B_{NL}}$ matrices for the LTI approximation model in order to calculate gains, that matrix is called

$$\breve{\boldsymbol{B}} = \boldsymbol{diag}(\breve{X}_u, \breve{Y}_v, \breve{Z}_w, \breve{K}_p, \breve{M}_q, \breve{N}_r), \tag{5.4}$$

and consists of characteristic drag terms linearized about expected operating conditions. Also the restoring forces and torques $\boldsymbol{g(\eta_2)}$ shall be omitted in the upcoming derivation for reasons explained in the next section. This reduces the 6DOF nonlinear governing equations model of Equation (2.14) to

$$\boldsymbol{\tau} = (\boldsymbol{M} + \boldsymbol{M_A})\dot{\boldsymbol{\nu}} + \breve{\boldsymbol{B}}\boldsymbol{\nu}, . \tag{5.5}$$

where $\boldsymbol{M}$ is the rigid body inertia matrix, $\boldsymbol{M_A}$ is the added mass matrix, $\dot{\boldsymbol{\nu}}$ are body frame accelerations, $\breve{\boldsymbol{B}}$ is the linearized characteristic damping matrix, and $\boldsymbol{\nu}$ are body frame velocities.

The symbol γ shall be used in this section to represent a new state variable vector $\boldsymbol{\gamma}$ as well as index notation where γ_i represents the i-th element of $\boldsymbol{\gamma}$. Redefining the conventional hydrodynamic terms with respect to different time derivatives of $\boldsymbol{\gamma}$ shall improve clarity of fundamental control law covered in this section. Let the state variable vector be

$$\gamma = \boldsymbol{J^{-1}_{1,\eta_2}}\boldsymbol{\eta}, \tag{5.6}$$

where $\boldsymbol{\gamma}$ represent 6DOF positions and orientations in the body-fixed reference frame. Let the first time derivative of the state variable vector be

$$\dot{\boldsymbol{\gamma}} = \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{J^{-1}_{1,\eta_2}}\dot{\boldsymbol{\eta}}_1 \\ \\ \boldsymbol{J^{-1}_{2,\eta_2}}\dot{\boldsymbol{\eta}}_2 \end{bmatrix} \tag{5.7}$$

where $\dot{\boldsymbol{\gamma}}$ is set equal to body-fixed velocities $\boldsymbol{\nu}$, Let the second time derivative of the state variable vector be

$$\ddot{\boldsymbol{\gamma}} = \dot{\boldsymbol{\nu}}. \tag{5.8}$$

The reason why new state variables are necessary is because desired states of the vehicle are best suited to be given in the inertial frame, however the gains and hydrodynamic model are provided in the body-fixed frame. Therefore, the 6DOF control law is most precisely

$$\boldsymbol{\tau}^* = \boldsymbol{k_P^*}\left(\boldsymbol{\eta_d} - \boldsymbol{\eta}\right) + \boldsymbol{k_D^*}\left(\dot{\boldsymbol{\eta}}_{\boldsymbol{d}} - \dot{\boldsymbol{\eta}}\right), \tag{5.9}$$

where $\boldsymbol{\tau}^*$ is some theoretical 6DOF vector of forces/torques in the inertial frame, $\boldsymbol{k_P^*}$ is the matrix of theoretical proportional gains for the inertial frame, $\boldsymbol{\eta_d}$ and $\dot{\boldsymbol{\eta}}_{\boldsymbol{d}}$ are desired positions/orientations and velocities in the inertial frame, $\boldsymbol{\eta}$ and $\dot{\boldsymbol{\eta}}$ are actual positions/orientations and velocities in the inertial frame, and $\boldsymbol{k_D^*}$ is the matrix of theoretical derivative gains for the inertial frame. The * superscript highlights the fact that these values are never explicitly determined, because $\boldsymbol{\tau}^*$ is immediately transformed to body-fixed frame. This makes the 6DOF control law in Equation (5.9) in the body-fixed frame

$$\boldsymbol{\tau} = \boldsymbol{k_P}\tilde{\boldsymbol{\gamma}} + \boldsymbol{k_D}\dot{\tilde{\boldsymbol{\gamma}}} - \boldsymbol{g}(\boldsymbol{\eta_2}) \tag{5.10}$$

where $\boldsymbol{k_P}$ is the diagonal matrix of proportional gains, $\tilde{\boldsymbol{\gamma}}$ is the vector of proportional errors, $\boldsymbol{k_D}$ is the diagonal matrix of derivative gains, and $\dot{\tilde{\boldsymbol{\gamma}}}$ is the vector of derivative errors.

Setting Equation (5.5) equal to Equation (5.9) converted to the body-fixed frame and replacing $\boldsymbol{\nu}$ terms with $\boldsymbol{\gamma}$

$$(\boldsymbol{M} + \boldsymbol{M_A})\ddot{\boldsymbol{\gamma}} + \breve{\boldsymbol{B}}\boldsymbol{\gamma} = \boldsymbol{k_P}\tilde{\boldsymbol{\gamma}} + \boldsymbol{k_D}\dot{\tilde{\boldsymbol{\gamma}}} \tag{5.11}$$

here $\boldsymbol{k_P}$ and $\boldsymbol{k_D}$ are gains for body-fixed frame errors Expanding the error terms

$$(\boldsymbol{M} + \boldsymbol{M_A})\dot{\boldsymbol{\gamma}} + \breve{\boldsymbol{B}}\boldsymbol{\gamma} = \boldsymbol{k_P}(\boldsymbol{\gamma_d} - \boldsymbol{\gamma}) + \boldsymbol{k_D}(\dot{\boldsymbol{\gamma}}_{\boldsymbol{d}} - \dot{\boldsymbol{\gamma}}), \tag{5.12}$$

then grouping desired state terms to one side and actual states to the other

$$\boldsymbol{k_P}\boldsymbol{\gamma_d} + \boldsymbol{k_D}\dot{\boldsymbol{\gamma}}_{\boldsymbol{d}} = \boldsymbol{k_P}\boldsymbol{\gamma} + (\boldsymbol{k_D} + \breve{\boldsymbol{B}})\dot{\boldsymbol{\gamma}} + (\boldsymbol{M} + \boldsymbol{M_A})\ddot{\boldsymbol{\gamma}}, \tag{5.13}$$

where the left-hand side is the forcing function dependent on gains and desired states and the right-hand side is the hydrodynamic model including the artificial forces/torques contributed by the 6DOF control law. In other words, the left-hand side is a particular function $f(t)$ and the right-hand side is a second-order system of the standard form $f(t) = k\boldsymbol{x} + d\dot{\boldsymbol{x}} + m\ddot{\boldsymbol{x}}$. This makes the parameters

$$k = \boldsymbol{k_P}; \quad d = \boldsymbol{k_D} + \breve{\boldsymbol{B}}; \quad m = \boldsymbol{M} + \boldsymbol{M_A}. \tag{5.14}$$

The natural frequency of a second-order LTI system is $\omega_n = \sqrt{k/m}$ which, after substitution, is defined for each degree of freedom as

$$(\boldsymbol{\omega_n})_{ii} = \sqrt{\frac{(\boldsymbol{k_P})_{ii}}{(\boldsymbol{M})_{ii} + (\boldsymbol{M_A})_{ii}}}. \tag{5.15}$$

In this paper, the notation $(\cdot)_{ii}$ refers to the i-th element of a diagonal matrix.

Linear time-invariant systems have solutions consisting of linear combinations of exponential functions of time. Characteristics of the exponential functions can be used to determine the behavior of the LTI system. For example, if any exponential function has a positive real exponent, the system response does not converge because it will grow unbounded. Given the special case of a second-order LTI system, a quantity known as the damping ratio, $\zeta = d/(2\omega_n m)$, may be used to characterize its behavior. If the value of $\zeta < 1$, the system is considered underdamped and will respond quickly, but overshoot and oscillate before converging. If the value of $\zeta > 1$ the system is considered overdamped and will respond slowly, and will not overshoot before eventually converging. If the value of $\zeta = 1$ the system is considered critically damped and will converge as quickly as possible without overshooting. We use it for the expression of the diagonal matrix whose elements are defined as

$$(\boldsymbol{\zeta})_{ii} = \frac{(\boldsymbol{k_D})_{ii} + (\breve{\boldsymbol{B}})_{ii}}{2(\boldsymbol{\omega_n})_{ii}(\boldsymbol{M} + (\boldsymbol{M_A})_{ii})}. \tag{5.16}$$

All of $\boldsymbol{M}$, $\boldsymbol{M_A}$, $\breve{\boldsymbol{B}}$, $\boldsymbol{\omega_d}$, $\boldsymbol{\zeta}$, $\boldsymbol{k_P}$ and $\boldsymbol{k_D}$ are diagonal matrices, which allows for the convenient notation and manipulations in (5.15) and (5.16). By setting the natural frequency ω_n to a desired value ω_d, we solve for proportional gain $\boldsymbol{k_P}$ with

$$\boldsymbol{k_P} = \boldsymbol{\omega_d}^2(\boldsymbol{M} + \boldsymbol{M_A}). \tag{5.17}$$

A critically damped system is optimal for the AR/VR pilot because it produces responsive control, while not being so fast to wobble and make the pilot nauseous. By substituting $\boldsymbol{k_P}$ into Equation (5.16) and setting $\boldsymbol{\zeta} = \boldsymbol{diag(1)}$ for a critically damped system, $\boldsymbol{k_D}$ becomes

$$\boldsymbol{k_D} = 2\boldsymbol{\omega_d}(\boldsymbol{M} + \boldsymbol{M_A}) - \breve{\boldsymbol{B}}. \tag{5.18}$$

To be clear, each of the six elements of the respective diagonal matrices $\boldsymbol{k_P}$ and $\boldsymbol{k_D}$ have different values for each of the six degrees of freedom. Within Equations (5.17) and (5.18), only the desired closed-loop natural frequency $\boldsymbol{\omega_d}$ is variable, with all other terms being constant; therefore, $\boldsymbol{\omega_d}$ acts as central dial which is set by the user, simultaneously determining proportional and derivative gains deciding the performance of the control law in Equation (5.10).

5.1.3 Complete Control Law and Feed-Forward Terms

In the previous section PD gains were calculated based on an LTI approximation of the full governing dynamics model from Equation 2.14 that neglected the restoring forces term $\boldsymbol{g(\eta_2)}$. If the control law were to ignore the restoring forces, there would be residual error. One solution to address the residual error is an additional integral term making the PD controller a PID controller which would accumulate a corrective force based on the residual error. This technique is not optimal because it would respond slowly to the residual error. Instead the chosen method for compensating for restoring forces is a feed-forward term appended to (5.10) that is nearly equal and opposite to $\boldsymbol{g(\eta_2)}$ as given by Equation (2.26). There is a slight modification that

$$\boldsymbol{g_C} = \begin{bmatrix} \boldsymbol{F_B} \\ \boldsymbol{r_b} \times \boldsymbol{J_{1,\eta_{2d}}^{-1}} \boldsymbol{B} \end{bmatrix} \tag{5.19}$$

where $\boldsymbol{g_C}$ is the 6DOF feed-forward control term to counteract restoring forces and torques. Given some “current” state and “future” desired state, the transformation matrix projects the feed-forward torque onto the desired angular orientation to actively drive the vehicle into that future orientation; whereas the feed-forward force is equal and opposite to the body-fixed net buoyancy force in order to passively nullify any perturbation caused by current linear restoring forces. Hence, the complete

6DOF control law is

$$\boldsymbol{\tau} = \boldsymbol{k_P}\tilde{\boldsymbol{\gamma}} + \boldsymbol{k_D}\dot{\tilde{\boldsymbol{\gamma}}} - \boldsymbol{g_C}. \tag{5.20}$$

5.2 General Tuning Methodology

The hybrid autopilot in mind for the BROV2 consists of decoupled control laws for heave, roll, pitch, and yaw. This decoupling simplifies the dynamics, drops the Coriolis/centripetal effects, and evaluates one degree of freedom individually as a starting point for controller design. The general approach to this process was to isolate one degree of freedom from the nonlinear governing equations model (Chapter 2, Equation 2.14) and determine its hydrodynamic coefficients with open-loop testing as described in Chapter 4. Those coefficients would then be updated in the simulation environment. Trim conditions based on the nonlinear drag curves and the operational expectations would determine the characteristic linearized drag for the LTI approximation. This would produce a PD control law based on the gains for a critically damped LTI system (Section 5.1.1), and serve as an initial starting point. A range of desired closed loop natural frequencies ω_d are then be tested in simulated step responses. The control law is then modified until the step response is comparable to the predicted critically damped step response. Next, the modified control law is tested in the water tank to observe the real-world response. If the real-world response does not meet expectations, the modified control law is adjusted iteratively between water tank tests. For more details on the implementation of the control law in ROS, see Appendix D.

5.3 Yaw Degree of Freedom

In this section, the procedure followed for tuning one degree of freedom, specifically the yaw direction, shall be reviewed. Because restoring forces are non-existent in yaw the equation for control forces in the yaw direction is

$$N_C = k_P(\omega_d)[\psi_d - \psi]_b + k_D(\omega_d)[\dot{\psi}_d - \dot{\psi}]_b, \tag{5.21}$$

which is the sixth element of the vector represented in Equation (5.20). Notice that there are no buoyant torques in the yaw direction, which simplifies calculations. In the case of control forces in the other degrees of freedom, an additional feed-forward term would appear because restoring

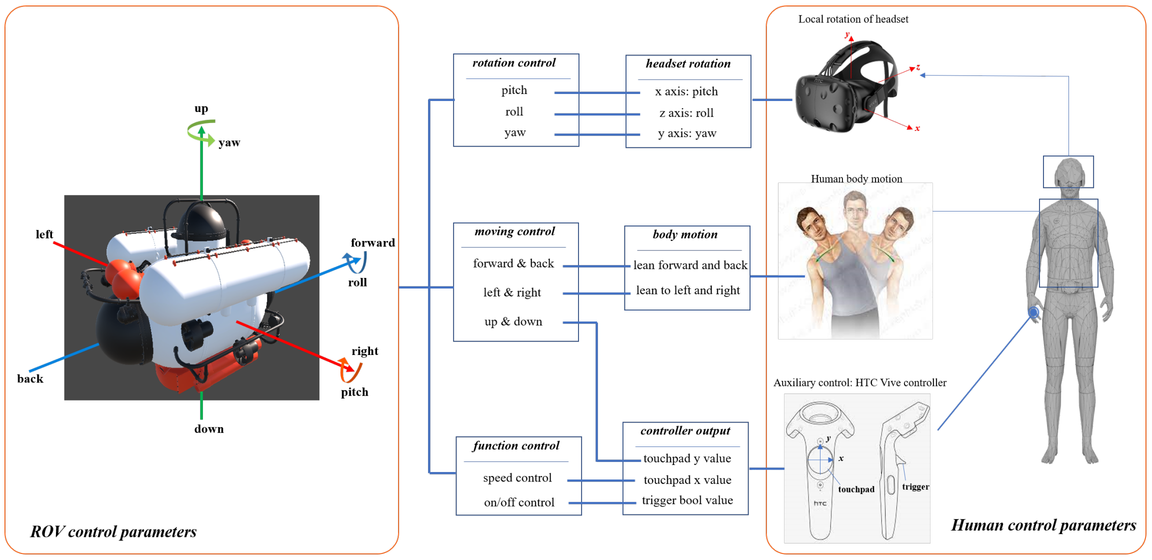


Figure 5.1: A chart showing the proposed gesture-based control for remotely operated vehicle (ROV) augmented/virtual reality (AR/VR) pilots proposed by collaborators from University of Florida (UF). Coordinate systems for the AR/VR pilot interface are Forward–Right–Down (FRD) body-fixed frame convention unique to Unity 3D development software [26].

forces/torques are nonzero. The lack of restoring forces/torques makes the yaw degree of freedom the best place to start calibrating the autopilot.

To give a high-level overview of how the control law works, Figure 5.2 shows a block diagram of the information flow throughout the portion of the hybrid autopilot that controls angular DOF. “Pilot Input” during the tests discussed in this section originated from prerecorded csvs outputting desired angular states $\boldsymbol{\eta_{2d}}$. These desired states are input into the “Control Law”, which uses gains calculated in Section 5.1 and produces the desired angular body torques $\boldsymbol{\tau_2}$. These torques are parsed with the “Thruster Allocation” block, implementing logic described in Section 3.4 and Algorithm 2 to determine thruster forces $\boldsymbol{m}$ that achieve the desired body torques. The collective effect of the eight thruster forces drive the “State/SITL” block, outputting the angular orientations $\boldsymbol{\eta_2}$ of the actual/virtual vehicle, respectively. Ultimately, the angular states are fed back to the “Control Law” block, establishing the closed loop of the angular controller.

Desired angular states are the preferred input for an AR pilot because of the gestures-based control that AR/VR pilots are planned to use (Figure 5.1). The VR headset naturally lends itself

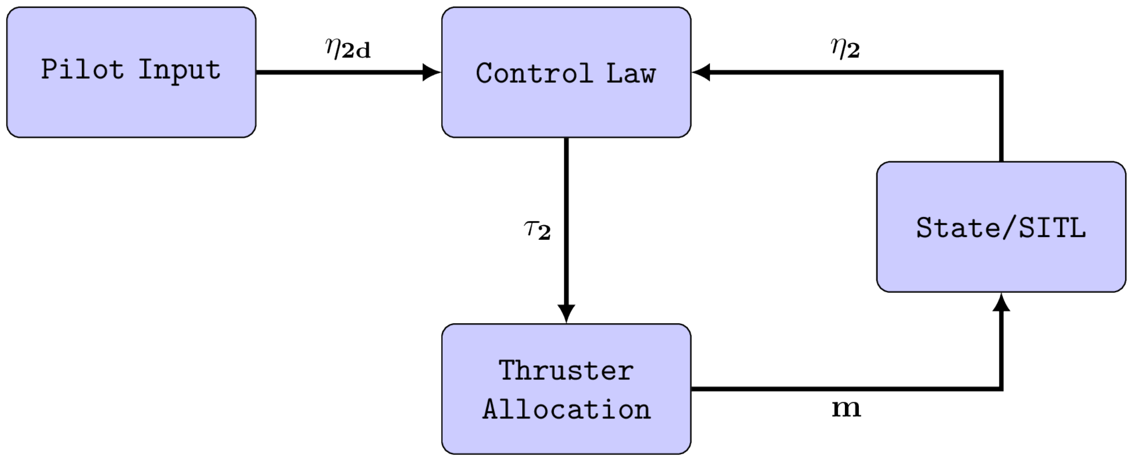


Figure 5.2: A block diagram of the information flow throughout the closed-loop portion of the hybrid autopilot that controls the angular degrees of freedom. Arrows are labeled with the important vectors that pass between the nodes.

to $\boldsymbol{\eta_{2d}}$, whereas if motions from the headset were not interpreted as desired states, but instead as open-loop jerking motions the pilot would become disoriented and uncomfortable very quickly.

In Section 5.1, the BROV2 is approximated as an LTI system in order to calculate control gains as a function of desired closed-loop natural frequency ω_d, but in actuality drag behaves nonlinearly, scaling quadratically with velocity. Therefore, an effective linear drag value $\breve{N}_r$ was determined with trim conditions, which was used to set the linear PD controller. The step response of the nonlinear system under Equation (5.21) was tested in SITL to verify behavior similar to a critically damped system.

Following the step response tuning, a series of frequency response tests documented in Appendix C were performed in SITL for low ($\omega_d = 2.0\,\text{rad/s}$), medium ($\omega_d = 3.0\,\text{rad/s}$), and high ($\omega_d = 4.0\,\text{rad/s}$) desired closed-loop natural frequency settings to produce a Bode plot. The final sequence of testing generalized the yaw control law across all angular degrees of freedom and activated the open-loop linear controls to test the performance of the total hybrid autopilot in STIL given a simulated dive involving all six degrees of freedom.

5.3.1 Determining Trim Conditions

While the standard technique to linearize the drag would be a first-order Taylor-series approximation at the center of the expected operational range of velocities, this would be unusable because the

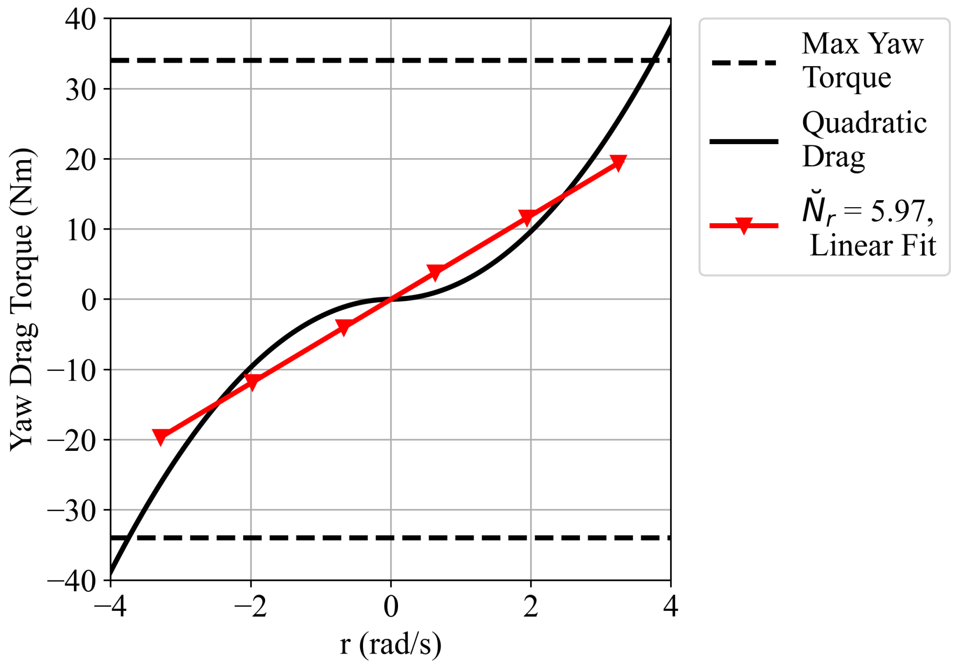


Figure 5.3: Nonlinear drag torque vs. velocity along with the trimmed linear fit based on $\breve{N}_r$. range.

line would not have any slope (since the velocity range is symmetric in both directions and the tangent to the nonlinear drag curve at $v = 0$ has zero slope). The idea was to choose a line trimmed about the origin so that there was no y-intercept allowing the slope to be used as $\breve{N}_r$. This trimmed line should approximate the expected drag torques as well as possible over the expected range of velocities. Considering that most ROV movements are typically low-amplitude minor adjustments, a value for $\breve{N}_r$ was selected that intersected the lower range of drag torques well. The linear fit of this approximation compared to the nonlinear yaw model shown in Figure 5.3.

5.3.2 Additional Damping Term to Correct for Nonlinear Behavior

With the effective linear damping approximation selected, step response simulations were performed and compared to the theoretical LTI system step response to assess how the controller handled a step input to rotate from a 0° heading to a heading of 90° and hold the position. A simple step input was generated such that at $t < 0$, $\psi_d = 0$ and at $t \geq 1$, $\psi_d = 90°$, with all other inputs null and put into comma-separated value (csv) file format. The step input was fed through a ROS `csv_reader` node into a separate ROS `control_law` node, which monitored vehicle state and calculated the force outputs according to the PD control law as described in Section 5.1.2. The resulting step responses were recorded to csv files with a ROS `logger` node (see Appendix D for more on nodes). This was then graphed against the theoretical step response.

The theoretical step response was calculated using the transfer function of the control law in the yaw direction which can be derived from Equation (5.13) by taking the Laplace transform and rearranging. The resulting transfer function is

$$H_L = \frac{\Psi(s)}{\Psi_d(s)} = \frac{k_P(\omega_d) + sk_D(\omega_d)}{s^2(I_{zz} + N_{\dot{r}}) + s(\breve{N}_r + k_D(\omega_d)) + k_P(\omega_d)} \tag{5.22}$$

where H_L is the closed-loop LTI transfer function, $\Psi(s)$ is the Laplace transform of yaw heading, $\Psi_d(s)$ is the Laplace transform of the desired yaw heading, k_P is proportional gain determined by the ω_d setting, k_D is derivative gain determined by the ω_d setting, and $\breve{N}_r$ is the chosen characteristic linear damping. Setting $\psi_d(t) = H(0)$, the Heaviside step function becomes $\Psi_d(s) = 1/s$ and performing the inverse Laplace transform on Equation (5.22), a function for $\psi(t)$ is obtained, which was used to generate the theoretical step response for various gains.

According to the linear function of the LTI system, the chosen ω_d and resulting k_P and k_D gains theoretically produce a critically damped response, which are shown as the dashed lines in Figure 5.4. Of note is the theoretical low $\omega_d = 2.0\,\text{rad/s}$ step response; there is an initial undershoot due to $k_D < 0$ causing the system to become a nonminimum-phase system [45]. The simulated response of the nonlinear system, shown with solid lines, produced underdamped step responses, that steadily improved with increasing ω_d, but did not reach critically damped behavior. As the goal is to attain critically damped behavior for the entire range of ω_d, the solution is to increase the damping via the artificial damping enacted by k_D. Observing how low ω_d needs more damping than higher ω_d, the amount of additional damping necessary is inversely related to ω_d. Hence, a small δk_D term was added to the original calculated k_D, which scales inversely with ω_d, with the form

$$\delta k_D = \frac{\kappa}{\omega_d^2} \tag{5.23}$$

where κ is a constant of proportionality. The ω_d term is squared because of the relatively low values of ω_d for this system. The new control law becomes

$$N_C = k_P(\omega_d)[\psi_d - \psi] + [k_D(\omega_d) + \delta k_D(\omega_d)][\dot{\psi}_d - \dot{\psi}], \tag{5.24}$$

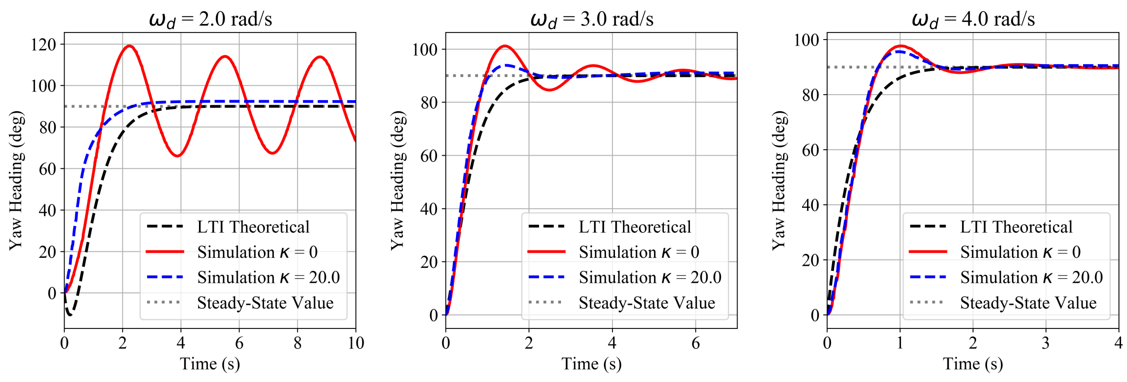


Figure 5.4: Step response testing in the nonlinear simulation. A tolerance of ±5% was needed to enable convergence because the thruster deadzone prevented slight adjustments when very close to the steady-state value of 90°.

A range of various constants of proportionality were tested across a wide range of ω_d. It was found that the value for the constant of proportionality that nullified oscillations best resulting in a critically damped or nearly critically damped step response was $\kappa = 20.0$. The resulting step response with the added δk_D term is shown in Figure 5.4 with dash–dotted lines.

It can be seen that with this additional term, the controller creates the desired behavior for the nonlinear vehicle system. At $\omega_d = 2.0\,\mathrm{rad/s}$, the $k_D + \delta k_D > 0$ modified derivative gain addresses the initial undershoot, and is large enough to prevent any overshoot having an even better settling time than the LTI theoretical settling time. For the $\omega_d = 3.0\,\mathrm{rad/s}$ and $\omega_d = 4.0\,\mathrm{rad/s}$ cases, there are slight overshoots, but they are acceptable for practical purposes.

A range of step response settling times of the nonlinear system using the tuned yaw controller of Equation (5.24) with δk_D resultant from $\kappa = 20.0$ was first performed in simulation. The desired closed loop natural frequencies tested were in $0.25\,\mathrm{rad/s}$ increments in the range of $1.5\,\mathrm{rad/s}, \leq \omega_d \leq 5.0\,\mathrm{rad/s}$ and compared with their theoretical linear settling times shown in Figure 5.5 with the blue dots. In the case of $\omega_d = 3.0\,\mathrm{rad/s}$, the overshoot falls within the ±5% tolerance and may be neglected. In the case of $\omega_d = 4.0\,\mathrm{rad/s}$, the initial overshoot exceeds the tolerance, but after one oscillation the step response converges; this is why there is a jump in the settling times for $\omega_d \geq 3.5$.

Also shown in Figure 5.5 are the settling times for the water tank step response tests at $\omega_d = \{2.0, 2.5, 3.5, 4.0, 4.5\}\,\mathrm{rad/s}$ also with δk_D resultant from $\kappa = 20.0$. Every ω_d was tested seven times each during water tank testing. Like the open-loop tests described in Section 4.1,

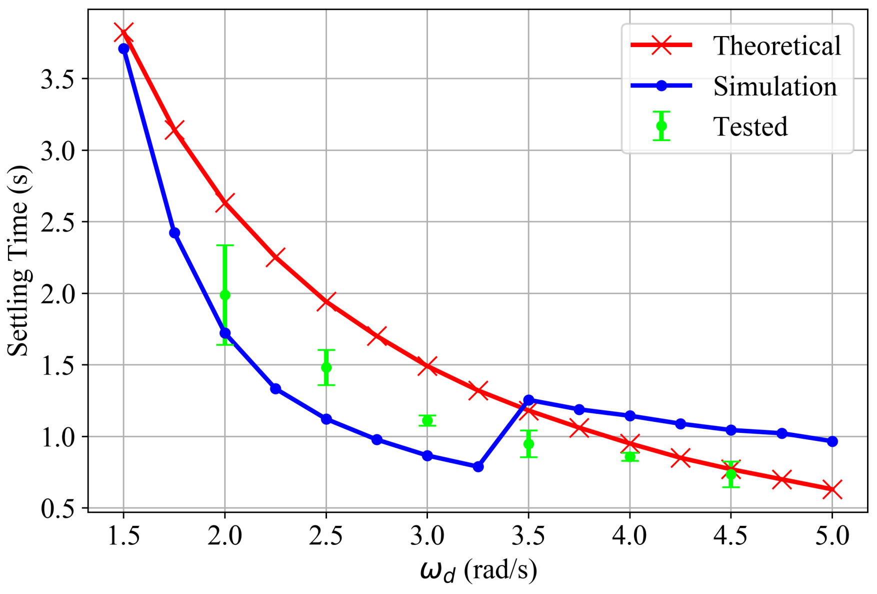


Figure 5.5: Graph comparing the theoretical, simulation, and experimentally tested settling times for a range of desired closed loop natural frequencies, ω_d. Settling time was defined as the amount of time it took the vehicle given a step input to stabilize within $\pm 5\%$ of the steady-state value of 90° yaw heading; results from the real-world test show error bars of two sample standard deviations.

each step response test was performed with the BROV2 nearly completely submerged, floating near the surface of the water, and initiated once while the BROV2 was totally at rest. Figure 5.5 demonstrates that the SITL simulation better approximates the actual response of the BROV2 than the LTI model, because the simulation predicts faster settling times at lower ω_d and the simulation results mirror this pattern. Settling times progress in a continuous fashion because they do not oscillate before converging; yet, there is a discontinuity in the simulation settling times because the SITL model, while taking into account hydrodynamic nonlinearity, does not account for thruster unsteady behavior and other physical complexities beyond the scope of this work. Tuned step responses of $\omega_d = \{2.0, 3.0, 4.0\}$ rad/s of the water tank experiments are shown with their accompanying simulation step responses in Figure 5.6. The $\omega_d = 2.0$ rad/s experimental response is nearly identical to the simulation; however, as ω_d increases the responses diverge slightly. This is in part due to the thruster ramp up time causing lag and otherwise resulting from imperfections in the model. It appears that the SITL model predicts sharper overshoots and faster convergence, which may be due to un-modeled higher dimensional drag terms.

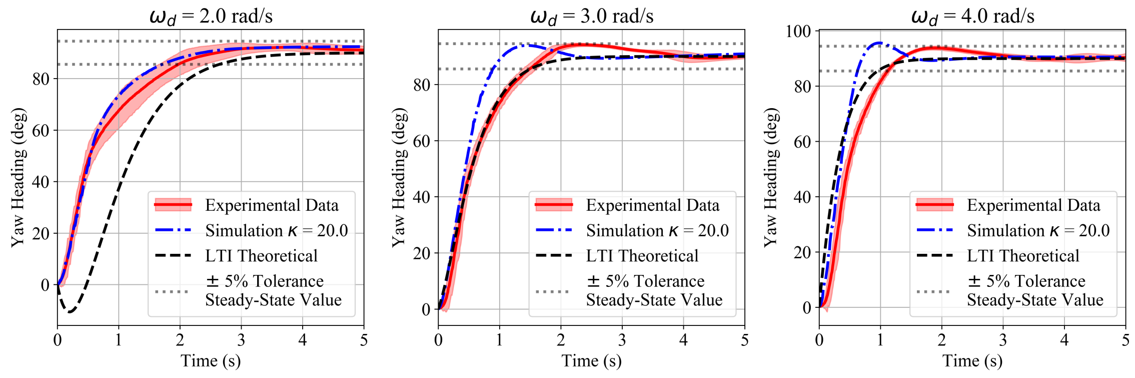


Figure 5.6: Three plots comparing tuned step responses in the water tank, using the updated software-in-the-loop (SITL) model, and predicted by LTI theoretical model (without additional damping). Experimental data from water tank step response testing are shown with a shaded region, where each vertical slice of time represents two sample standard deviations around the average response of seven trials; also plotted are simulation results and linear approximation predictions.

Despite these shortcomings, the updated simulation model is a vast improvement over the old simulation model, as can be seen in Figure 5.7. The old hydrodynamic model predicts a much slower step response than what was actually observed during the water tank experiments. This is due to the numerous errors and crude approximations in the originally available simulation model. Further quantification is presented in Table 5.1, where the average normalized error of the simulated step responses of the old and new SITL models are compared to what was observed experimentally across a range of ω_d settings. Average normalized error, $\boldsymbol{\overline{e}}$, is defined as

$$\overline{e} = \frac{1}{T}\int_0^T \frac{|e|}{\psi}dt, \tag{5.25}$$

where T is the length of time that the averaging interval is measured, $|e|$ is the absolute error between the simulated and experimental heading, ψ is the experimental heading, and t represents time, of which both $|e|$ and ψ are functions. The columns labeled $\boldsymbol{\overline{e}_{old}}$ and $\boldsymbol{\overline{e}_{new}}$ characterize the error of the old and new SITL models, respectively. It can be seen in Table 5.1 that the old model predicts significantly more errors than the our updated model. Figure 5.7 also displays that the updated model captures the higher order behavior well. This proves that the methodologies for system identification described in Section 4.1 and controller tuning described in Section 5.1 are effective tools for designing BROV2 performance.

Table 5.1: Average normalized error of old and new software-in-the-loop (SITL) models with respect to experimental step responses for five-second window [32].

ω_d (rad/s)	$\overline{e}_{old}$ (deg)	$\overline{e}_{new}$ (deg)
2.0	63.80	3.66
2.5	45.53	8.26
3.0	37.53	5.20
3.5	34.18	4.32
4.0	26.36	5.27
4.5	24.60	4.10

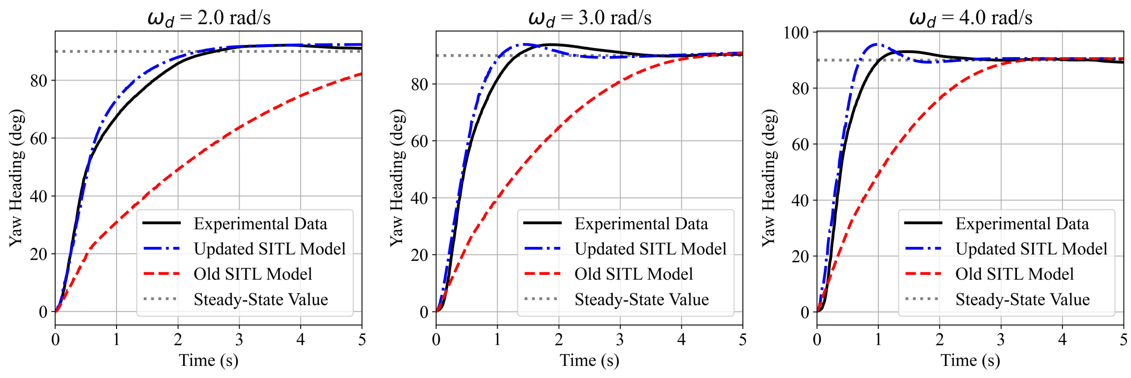


Figure 5.7: Graphs of tuned step responses comparing simulation results of the original SITL hydrodynamic model, the new SITL model from this work, and the water tank results.

5.4 Simulated Pilot Testing

The final set of testing used a simulated pilot dive containing all 6DOF provided by collaborators from the UF. Like the previous tests, inputs were parsed by a `csv_reader` ROS node and logged with a `logger` node. What was different for this test was that the `control_law` node was extended to include all three angular degrees of freedom,

$$\begin{bmatrix} K_C \\ M_C \\ N_C \end{bmatrix} = \boldsymbol{k_P J_{2,\eta_2}^{-1}}(\tilde{\boldsymbol{\eta}}_2) + \boldsymbol{k_D J_{2,\eta_2}^{-1}}(\dot{\tilde{\boldsymbol{\eta}}}_2) - \boldsymbol{g_{C,4:6}}. \tag{5.26}$$

The equations for K_C and M_C are identical to Equation (5.21) except for additional feed-forward buoyancy terms that were discussed in Section 5.1.3. The presence of the feed-forward terms directly counteract the hydrostatic righting moment at the desired orientation, preventing persistent steady-state error due to false convergence. As the simulation and control law models for roll and pitch are identical to yaw, nonminimum phase behavior at low ω_d and underdamped behavior were observed, so $\delta k_D = \kappa/\omega_d^2$, with $\kappa = 20$, was also applied.

Linear controls in the surge, sway, and heave directions were treated like direct inputs from a joystick and did not have error feedback control laws associated with them. This meant that the magnitude of a linear signal required the percentage of the associated elements of the maximum forces and torques (see Appendix B). The following results of the orientation trajectory are shown row-wise for $\omega_d = \{2.0, 3.0, 4.0\}$ rad/s in Figure 5.8. For every increase of ω_d, there are marginal improvements in satisfying the desired trajectories, most apparent in the yaw column. To represent these improvements more clearly, Figure 5.9 displays the integral of the absolute error between the desired and actual trajectories. Of note is how the pitch error does not improve significantly by increasing ω_d from 2.0 rad/s to 3.0 rad/s. This could be a result of managing buoyancy stability while carrying out challenging roll commands simultaneously. In general, it can be seen that error decreases with increasing ω_d.

Figure 5.10 shows the linear open-loop command signals of the simulated dive in the top row, and graphs representative of the energy costs of the angular controllers are in the lower row. The values for the energy costs are summed and normalized to the PWM signal from their respective

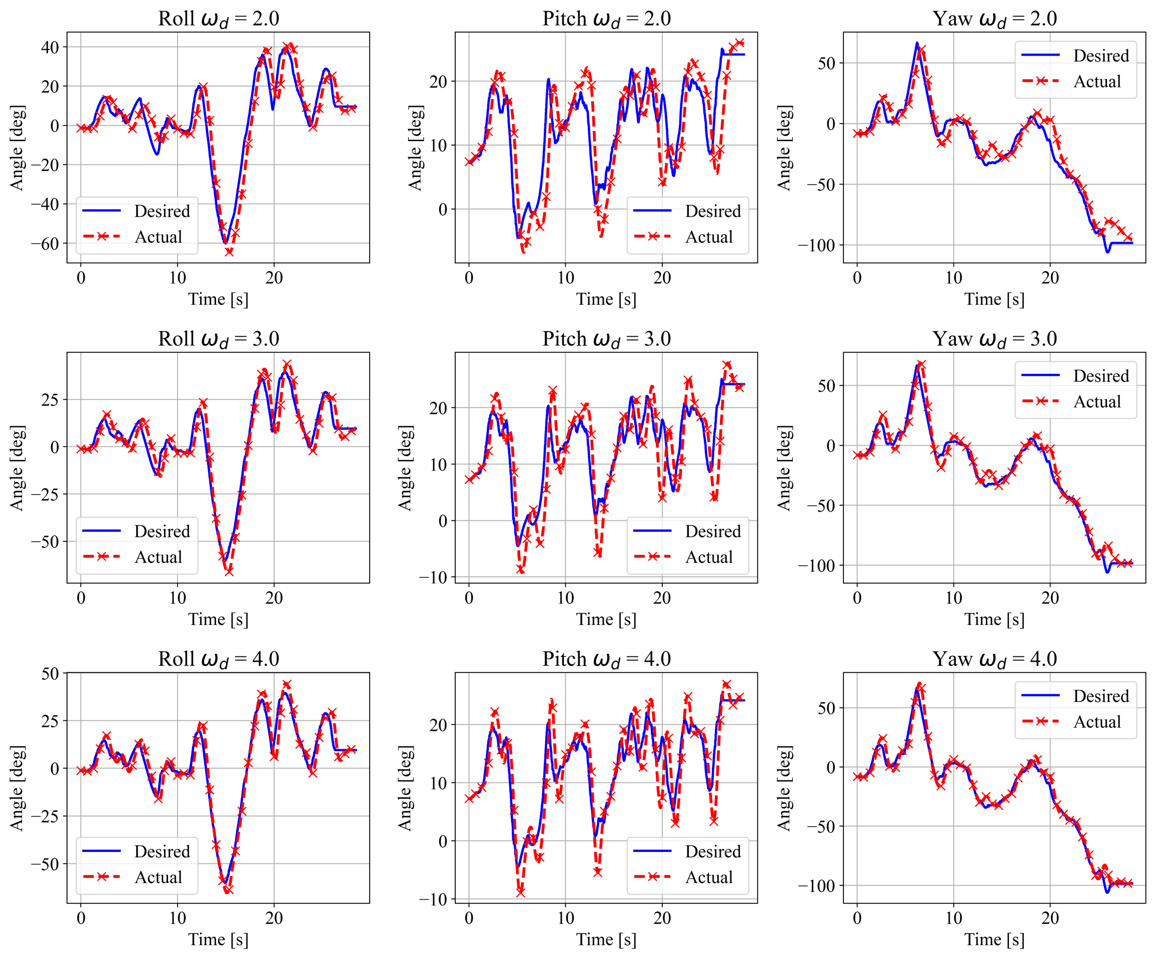


Figure 5.8: Plots comparing desired and actual trajectories of the simulated dive. Each row shows a different ω_d; every column displays a different angular degree of freedom.

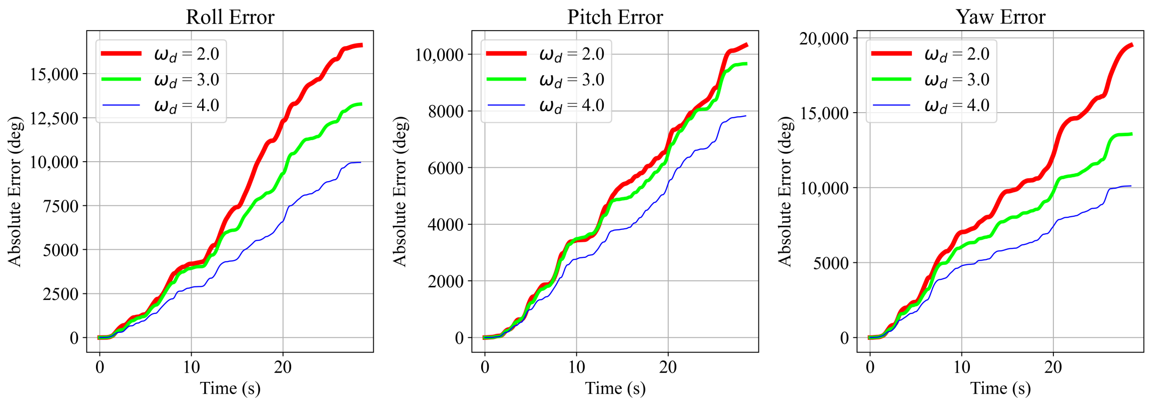


Figure 5.9: Shown are the integrals of absolute error with respect to time at the different ω_d. The same general pattern of error is present for all ω_d, but there are decreases in magnitude with increases in ω_d.

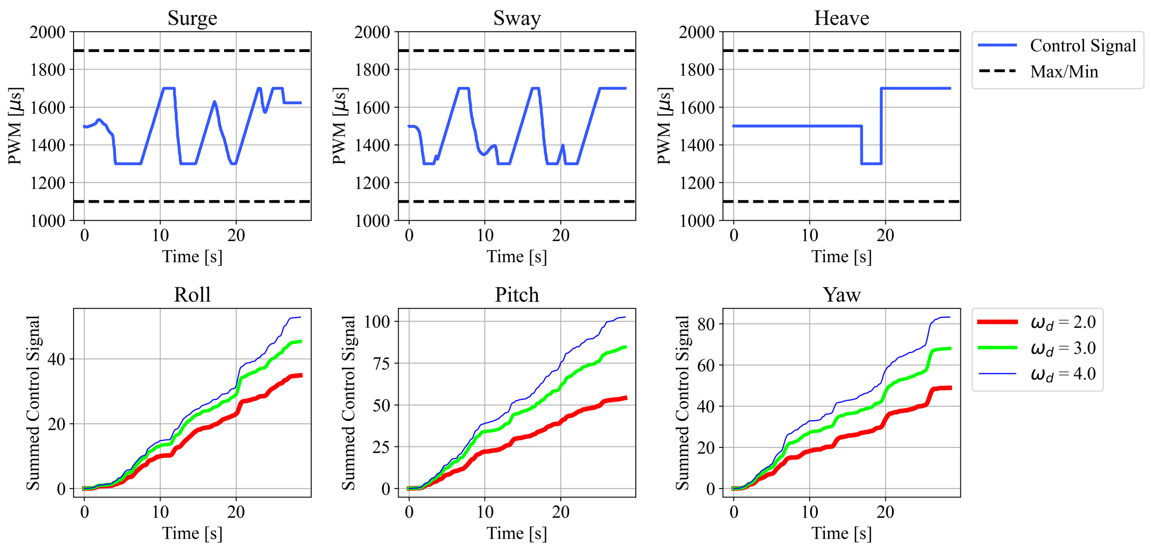


Figure 5.10: Graphs displaying linear commands and energy expenditure. First row shows the open-loop control PWMs of linear degrees of freedom with respect to time, and second row shows summed PWMs of angular degrees of freedom defined in Equation (5.27), correlating energy costs of different ω_d.

controllers or

$$\sum_{i=1}^{n}\left(\frac{|PWM - 1500|}{400}\right) \tag{5.27}$$

where PWM is the control signal sent for the respective degree of freedom, i is the index of the autopilot update, and n is the total autopilot iterations of the dive. An absolute value is taken in the numerator to group the magnitudes of positive and negative control signals because the PWM signals are centered on 1500 µs. This value is normalized with the maximum range ±400 µs of control signals. It can be seen that energy costs increase with increasing ω_d.

As a whole, the hybrid autopilot performed very well in tracking the angular states prescribed by the simulated pilot dive. A noticeable improvement in pitch error in ω_d, changing from 3.0 rad/s to 4.0 rad/s, indicates the nuance in selecting appropriate hybrid autopilot settings that the pilot must grow accustomed to. Energy costs are also another consideration that should be considered by pilots when planning their missions. Broadly speaking, the BROV2's performance will depend in equal parts on the choice of ω_d and the pilot's handling via the AR/VR interface.

Chapter 6

Conclusion

Work done to develop an autopilot for the BROV2 Heavy was the subject of this thesis. The BROV2 is a popular choice for researchers because it is affordable and encourages customization because it is built with open-source hardware, software, and firmware. This vehicle was chosen to demonstrate that an AR/VR interface for pilots would be beneficial to accelerate workforce training and encourage more widespread use of ROVs. It was decided that a hybrid autopilot, which combined both error feedback and non-error feedback modes of control, would assist AR/VR pilots by interpreting gestures as inputs that manipulate the vehicle. The research group at UHM was tasked with developing this hybrid autopilot for the BROV2. In doing so, a more general autopilot following LTI control principles was developed to be used modularly through ROS.

The first topic of this thesis, following the introduction of the governing dynamics, was solving the practical issue of quantitative control for the BROV2. ArduSub is a general autopilot for marine submersibles and did not have this ability standard. In order to have precise automatic abilities, quantitative control is necessary. Customization to the firmware implemented the thruster allocation matrix of the BROV2 Heavy and a lookup table for the individual thrusters. With these updates the vehicle is now able to receive desired 6DOF forces and torques and fulfill them with precision, without needing any special pilot inputs.

Open-loop water tank testing was performed to determine values for the hydrodynamic coefficients. Experiments were performed that analyzed one degree of freedom at a time, simplifying the coupling effects of centripetal forces, and mirroring the techniques used to calculate control gains later. The yaw degree of freedom was characterized first. Drag effects were able to be isolated because constant yaw velocity was able to be reached. Inertia effects in yaw were then determined by studying the unsteady regime while the vehicle was still accelerating. The heave degree of freedom was characterized next, but the testing water tank was not tall enough for constant heave velocity to be achieved. It was observed that rise and dive responses had distinctive differences due to the significant asymmetry between the upper and lower halves of the BROV2 Hence, a piecewise model for heave was proposed. Least squares analysis was used to fit drag and inertia terms simultaneously for the open-loop rise and dive data separately. Unknown model coefficients were scaled proportionally to the experimentally determined yaw and heave models for remaining angular and linear degrees of freedom, respectively, at ratios suggested by other research concerning the BROV2. Standard ArduSub SITL does have a simulation model of the BROV2, but it was very inaccurate. The 6DOF model coefficients updated the SITL model and are also used to determine control gains.

A mathematical procedure for determining the foundational control law was outlined. This method uses an LTI approximation to determine control gains. It involved choosing some characteristic linearized drag term to approximate the nonlinear drag. Only one degree of freedom is treated at a time so the coupling due to centripetal effects are ignored. Using the linear approximation, PD gains for a critically damped system are generated as a function of desired closed loop natural frequency ω_d. It was observed in SITL that the foundational yaw controller model produced an underdamped response that grew less apparent as ω_d increased due to the limitations of the linearized approximation for a nonlinear system. Thus, after more SITL testing it was found that an additional damping term that was inversely proportional to ω_d^2 calibrated the foundational yaw controller model to prevent overshoot and oscillations. Water tank testing confirmed settling times of the tuned yaw control law comparable to those predicted in simulation, validating the effectiveness of the updated SITL model.

The ArduSub SITL shall continue to serve as a test bench for future development of the autopilot features. Immediate future work will involve water tank testing of step responses of heave following a similar approach as yaw. The added complexity of the piecewise model presents an opportunity to try different strategies for modifying the foundational heave controller model, like gains scheduling, for example. The BROV2 would switch its PD gains based on its state; exactly what the switching conditions should optimally be is unknown. Once robust depth control is achieved, data will be collected for the roll and pitch degrees of freedom to determine their accurate hydrodynamic models. Coupling between the heave direction and those motions would be impossible to ignore, however, because vertical thrusters would have to be used to counteract restoring forces and provide command torques, presenting another set of challenges. Continued work on the AR/VR project will involve discussion on how the hybrid autopilot can best complement the pilot. This may involve more ROS modules to be developed or mixing the existing ones in new ways. The modular capabilities of ROS in conjunction with the quantitative firmware-level control also open the path towards exploring nonlinear control strategies.

Appendix A

Hardware and Software Considerations

The contents of this appendix directly quote from [32].

Out of the box, the R1, R2, and R3 versions of the BROV2 come equipped with a Pixhawk and the R4 version (those sold after 7 June 2023) comes with a Navigator [46], both of which are flight management units (FMUs). Regarding hardware connections, onboard the FMU are input and output serial ports, sensors, and pulse width modulation (PWM) output connections for accessories and motor electronic speed controllers (ESCs). Onboard the R1-R3 version, the Pixhawk FMU handles the pilot inputs, sensor processing, and state estimation on a STM32F427 system on a chip (SoC). The Pixhawk SoC has a 180 MHz CPU, 256 KB of RAM, and a 2 MB flash drive which the ArduSub firmware is stored on [47].

Navigator FMUs on R4 versions act as a hardware-on-top augmentation of the Raspberry Pi 4 companion computer, giving FMU computations direct access to the much more powerful single-board computer. Perhaps in consideration of the lean technical specifications of the Pixhawk and other lightweight FMUs that may be utilized on marine vehicles, the ArduSub source code was written with efficiency in mind [43]. That is to say, certain software limitations stem from hardware limitations. The autopilot libraries indicated with the '`AP_`' filename prefix are built using custom math functions and primitive data types, performing the bulk of computations with three element vectors, three-by-three matrices, and a limited selection of linear algebra operations. While this implements an Extended Kalman Filter on a Pixhawk, it must do so in roundabout ways to circumvent a more extensive matrix math library that would require larger processing overhead. In the ArduSub source code, this manifests as frequently breaking apart calculations into smaller pieces, like separating linear and angular dimensions into two three-dimension vectors, using program loops that iterate over the elements, and then recombining pertinent values with summation and scaling.

The key challenge in developing ArduSub is operating within the existing software framework because external libraries cannot be used.

Appendix B

Implementation of Thruster Allocation Matrix

The contents of this appendix directly quote findings from [32].

In implementing custom changes to ArduSub, it is important that modifications are precise and that the rest of the codebase's functionality is preserved. The locations in the code handling individual thruster allocation were identified to be within the files `SIM_Submarine.cpp` and `AP_Motors6DOF.cpp`. The former carries out calculations for state updates of the software-in-the-loop (SITL) virtual vehicle due to motor inputs and contains a matrix

$$\widehat{\boldsymbol{T}} = \begin{bmatrix} -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{B.1}$$

Here, a hat is used to distinguish this matrix from the matrix $\boldsymbol{T}$. The latter contains a matrix for distributing pilot control inputs to the individual thruster activations with the form

$$\widehat{\boldsymbol{T}}^{\top} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 1 \\ -1 & -1 & 0 & 0 & 0 & -1 \\ 1 & 1 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 1 & 0 \end{bmatrix}, \tag{B.2}$$

where $\widehat{\boldsymbol{T}}^{\top}$ is the transpose of Equation (B.1). Upon closer inspection of the two matrices, both are organized in the same fashion as the thruster allocation matrix and its inverse, and are used for similar purposes: $\widehat{\boldsymbol{T}}$ is to $\boldsymbol{T}$ as $\widehat{\boldsymbol{T}}^{\top}$ is to $\boldsymbol{T}^{\dagger}$. Where the zero entries in $\widehat{\boldsymbol{T}}$ and $\widehat{\boldsymbol{T}}^{\top}$ are the entries close to zero in $\boldsymbol{T}$ and $\boldsymbol{T}^{\dagger}$, the nonzero entries also share the same positive or negative sign. However, the hatted matrices lack the dimensionality that the theoretical ones possess because they do not implicitly encode physical units and were intended for a heuristic control scheme instead of a quantitative one.

Figure B.1 provides a simple flow chart visualization of the following description. Every iteration of the software control loop checks the `RC_Input` channel for the six-by-one vector of PWM signals ranging from 1100 µs to 1900 µs, centered on 1500 µs where values $>$ 1500 µs are positive values and $<$ 1500 µs are negative values. The six entries correspond to surge, sway, heave, roll, pitch, and yaw commands and are each normalized to a -1.0 to 1.0 range that becomes the user input vector $\boldsymbol{u}$. This vector $\boldsymbol{u}$ is then read into `AP_Motors6DOF.cpp` where the function `motor_command()` calculates the thruster allocation; Algorithm 1 details this loop and shall be discussed shortly.

Calculations performed in the primary loop of `AP_Motors6DOF.cpp` take the forces and torques requested by $\boldsymbol{u}$ and determine how the eight individual thrusters should activate to serve these commands. The resulting eight-by-one vector of these calculations $\boldsymbol{m}^{*}$ is analogous to the forces from Equation (3.5). Each entry represents one of the eight thrusters and is normalized to a -1.0 to 1.0 range. The entries of $\boldsymbol{m}^{*}$ are converted once more to PWM signals in the 1100 µs to 1900 µs range and output as $\widehat{\boldsymbol{m}}$ to the `RC_Output` channel. Once in `RC_Output`, the PWMs are distributed

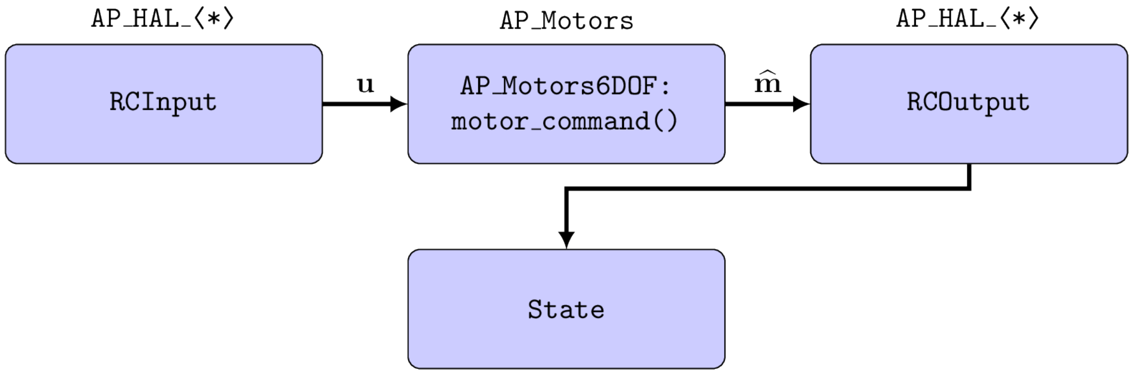


Figure B.1: Flowchart of data flow between ArduSub/Ardupilot programs. Manual control data travel through programs sequentially in arrays which are marked with arrows below their symbolic variables; above the boxes are the library names and the inside of the boxes are the module names.

throughout the respective ESCs corresponding to the eight individual thrusters. Ultimately, this activates the thrusters that generate forces and torques on the vehicle.

To understand how exactly $\widehat{\boldsymbol{T}}^\top$ is used, Algorithm 1 provides a step-by-step clarification. As of the ArduSub 4.5.0 beta1 release on 22 February 2024, the Ardupilot/ArduSub source code still utilizes Algorithm 1 found in `AP_Motors6DOF.cpp` for the manual control mode [48]. The matrix is used as a constant parameter that helps transform user input $\boldsymbol{u}$ into motor activation $\boldsymbol{m}^*$. The function performing the computation is `motor_command()`. Within the function, there are various forms of ξ which serve as buffer variables used inside the function. The first usage is introduced on lines 2 and 9, with the form ${}^{(j)}\xi_{max}$, where the left superscript j can be either 1 for roll, pitch, and heave control or 2 for yaw, surge, and sway control and the right subscript "max" labels its purpose as a normalizing variable. The second type of usage is introduced on lines 4 and 11, with the form ${}^{(j)}\xi_i$, where the left superscript j again can be either 1 for roll, pitch, and heave control or 2 for yaw, surge, and sway control and the right subscript i is an index indicating the i-th thruster which is being calculated for. The groupings of forces in ${}^{(1)}\xi$ terms correspond to those generated primarily by the vertical thrusters and groupings of forces in ${}^{(2)}\xi$ correspond to those generated primarily by the azimuthal thrusters.

The function `motor_command()` consists of three loops, two for calculating preliminary output values based on inputs and the third loop for constructing the output. The first loop and second loop are identical except that the first loop calculates for roll, pitch, and heave control and the second

Algorithm 1: Original Manual Control Algorithm

```
   Parameters: $\widehat{\boldsymbol{T}}^{\top}_{\mathbf{8\times 6}}$
   Input: $\boldsymbol{u}_{\mathbf{6\times 1}}$
   Output: $\boldsymbol{m}^{*}_{\mathbf{8\times 1}}$
1  Function motor_command($\boldsymbol{u}$,$\widehat{\boldsymbol{T}}^{\top}$):
2      ${}^{(1)}\xi_{max} = 1$;
3      for $i \leftarrow 1$ to $8$ do
4          ${}^{(1)}\xi_i = u_3 \cdot \widehat{T}^{\top}_{i,3} + \boldsymbol{u}_{4:5} \cdot \widehat{\boldsymbol{T}}^{\top}_{i,4:5}$;
5          if $|{}^{(1)}\xi_i| >^{(1)} \xi_{max}$ then
6              ${}^{(1)}\xi_{max} =^{(1)} \xi_i$;
7          end
8      end
9      ${}^{(2)}\xi_{max} = 1$;
10     for $i \leftarrow 1$ to $8$ do
11         ${}^{(2)}\xi_i = \boldsymbol{u}_{1:2} \cdot \widehat{\boldsymbol{T}}^{\top}_{i,1:2} + u_6 \cdot \widehat{T}^{\top}_{i,6}$;
12         if $|{}^{(2)}\xi_i| >^{(2)} \xi_{max}$ then
13             ${}^{(2)}\xi_{max} =^{(2)} \xi_i$;
14         end
15     end
16     for $i \leftarrow 1$ to $8$ do
17         $m^*_i = \frac{{}^{(1)}\xi_i}{{}^{(1)}\xi_{max}} + \frac{{}^{(2)}\xi_i}{{}^{(2)}\xi_{max}}$;
18         constrain($m_i \in [-1.0, 1.0]$);
19         $\widehat{m}_i = 1500 + 400m^*_i$;
20     end
21     send $\widehat{\boldsymbol{m}}$ to RC_Output channels;
```

loop calculates for yaw, surge, and sway control. The first two loops step through the thrusters in order and scale outputs with respect to the corresponding parts of $\widehat{\boldsymbol{T}}^{\top}$ with respect to the user inputs $\boldsymbol{u}$ and sum them together in the buffer ${}^{(j)}\xi_i$ (lines 4 and 11). These numbers are then checked against the current value of ${}^{(j)}\xi_{max}$ and replaced with the value of the ${}^{(j)}\xi_i$ buffer if it is larger (lines 5 and 12). After all eight thrusters have been looped over twice, the final third loop normalizes ${}^{(j)}\xi_i$ with respect to the final values of ${}^{(j)}\xi_{max}$ and adds the vertical and azimuthal ratios together (line 17). The trimming function `constrain()` checks that each value of $\boldsymbol{m}^*$ is in the range of -1.0 to 1.0 (line 18); if a value is outside the bounds, the function chooses the limit closest. Then, the values of $\boldsymbol{m}^*$ are scaled up to appropriate PWMs (line 19), and finally sent to `RC_Output` which relays the signals to the respective ESCs.

If a user were to command any combination of movement, the vehicle would respond with all requested movements, but all would be scaled with respect to the largest commanded force or torque. The overall effect of Algorithm 1 is that it generates intuitive vehicle responses that a pilot will recognize as matching joystick inputs. This heuristic method is a good way to prevent cases of motor saturation where the vehicle appears to be unresponsive due to motors being maxed out. However, the exact value of the resulting forces/torques, $\boldsymbol{\tau}$, will not directly correlate to the magnitude of the input commands, which is necessary for automatic control laws. Instead, quantification of forces should be preserved, and saturation avoided through selection of an adequate control bandwidth ((which is proportional to the desired natural frequency ω_d).

Thus, with this assessment of the original control implementation and the goal of implementing a PD control law, a new simple algorithm is determined which can replace the pre-existing framework without disrupting anything else or introducing unnecessary complexity.

In order for the new manual control algorithm to be integrated seamlessly, without altering processes on either end of the software control loop, it must use the same inputs and produce the same outputs. The main challenge is incorporating physical units that will allow the autopilot to command specific quantities of forces and torques. To achieve this, $\boldsymbol{T}^{\dagger}$ replaces $\widehat{\boldsymbol{T}}^{\top}$ and new parameters and functions are introduced that encode the BROV2 hardware limitations while preserving the control architecture. These new parameters are $\boldsymbol{\tau_{max}}$, $\boldsymbol{PWM^*}$, and $\boldsymbol{t}$, which are a six-by-one vector containing the maximum forces and torques that the BROV2 is capable of, eighty-one normalized PWMs and eighty-one corresponding thrusts produced by a T200 thruster. The contents of $\boldsymbol{PWM^*}$ and $\boldsymbol{t}$ are experimental data collected by Blue Robotics [40] and graphed in Figure 3.3 and used as a lookup table, utilized by the FMU thrust distribution and the SITL hydrodynamic model. The values of $\boldsymbol{\tau_{max}}$ are determined analytically by solving for $\boldsymbol{\tau}$ using Equation (3.1) given $\boldsymbol{m}$, constructed of maximum thrusts according to the lookup table in combinations producing solely surge, sway, heave, roll, pitch, or yaw maneuvers. This results in $\boldsymbol{\tau_{max,1:3}} = [127.32, 127.32, 199.81]\,\mathrm{N}$ for forces and $\boldsymbol{\tau_{max,4:6}} = [39.25, 21.61, 34.00]\,\mathrm{Nm}$ for torques. To use input signals from `RC_Input` of $\boldsymbol{u} \in [-1.0, 1.0]$, the signals are converted to units of forces and torques with $\boldsymbol{\tau_{max}}$, and transformed with $\boldsymbol{T}^{\dagger}$ to find $\boldsymbol{m}$ directly. Mathematically, these operations are

$$\boldsymbol{m} = \boldsymbol{T}^{\dagger}\boldsymbol{diag}(\boldsymbol{\tau_{max}})\boldsymbol{u}. \tag{B.3}$$

Algorithm 2 shows how Equation (B.3) is accomplished in ArduSub step by step. Using the new parameters $\boldsymbol{\tau_{max}}$, $\boldsymbol{PWM^{*}}$, and $\boldsymbol{t}$, the same inputs $\boldsymbol{u}$ are used and the same outputs $\widehat{\boldsymbol{m}}$ are produced. However, now $\boldsymbol{u}$ originates from a hybrid autopilot which has calculated physical forces and torques (that have been normalized with respect to $\boldsymbol{\tau_{max}}$), as opposed to interpreting inputs heuristically. Linear DOF control is handled with an open-loop controller that directly passes inputs into $\boldsymbol{u_{1:3}}$. Angular control is handled with a closed-loop PD control law that interprets an AR pilot's gestures with error feedback, before passing inputs into $\boldsymbol{u_{4:6}}$.

The new function `motor_command_new()` uses one software loop; first, it iteratively performs Equation (B.3) (line 3) to determine m_i. This is then input into the function `find_bounding_indicies()`, which determines the closest indices of $\boldsymbol{t}$ below $(_{low})$ and above $(_{high})$ the force of m_i (line 4). In the event that m_i is outside the range of $\boldsymbol{t}$, the $_{low}$ index will equal the $_{high}$ index and default to either -1.0 for negative forces or 1.0 for positive forces (lines 5 through 12). For all other instances, the bounding indices will be used to perform linear interpolation that produces a normalized value of m_i^{*} (lines 13 through 15). This value is trimmed (line 16) and then converted to PWMs (line 18) to finally be sent to `RC_Output`. It is important to emphasize that while this new algorithm accounts for vehicle limitations, it depends on inputs from the hybrid autopilot that encode vehicle limitations as well.

Algorithm 2: New Manual Control Algorithm

```
Parameters: $\boldsymbol{T}^{\dagger}_{\boldsymbol{8\times 6}}, \boldsymbol{\tau_{max,8\times 1}}, \boldsymbol{PWM^*_{81\times 1}}, \boldsymbol{t_{81\times 2}}$
Input: $\boldsymbol{u_{6\times 1}}$
Output: $\boldsymbol{m^*_{8\times 1}}$
Function motor_command_new($\boldsymbol{u}$,$\boldsymbol{T}^{\dagger}$, $\boldsymbol{\tau_{max}}$):
    for $i \leftarrow 1$ to $8$ do
        $m_i = \sum_{j=1}^{6} \left(T^{\dagger}{}_{i,j} \cdot \tau_{max,j} \cdot u_j\right)$;
        $_{low}, _{high}$ = find_bounding_indices($m_i$);
        if $_{low} =_{high}$ then
            if $m_i < 0$ then
                $m_i^* = -1.0$
            end
            else if $m_i > 0$ then
                $m_i^* = 1.0$
            end
        end
        else
            $m_i^* = PWM^*_{low} + \frac{PWM^*_{high} - PWM^*_{low}}{t_{high} - t_{low}}(m_i - t_{low})$;
        end
        constrain($m_i \in [-1.0, 1.0]$);
        $\widehat{m}_i = 1500 + 400 m_i^*$;
    end
    send $\widehat{\boldsymbol{m}}$ to RC_Output channels;
```

Appendix C

Frequency Response Testing

The contents of this appendix directly quote [32].

After achieving effectively critically damped step responses, the next sequence of testing for characterizing the performance of the yaw PD control law was closed-loop frequency response testing with SITL. Observations were made of the vehicle's ability to follow a $\pm 30^\circ$ desired yaw heading sine wave. A set of desired frequency yaw trajectories were generated from 0.05 Hz to 1.97 Hz in 0.04 Hz increments and saved as csv files. These trajectories were fed into the ROS `csv_reader` node which read the csv line-by-line in sync with the simulation such that the system had no knowledge of future headings (similar to the AR pilot commands). A separate ROS `control_law` node monitored vehicle state and calculated the force outputs according to the modified PD control law as described in Section 5.3.2 with the addition of δk_d, as discussed in the previous section. The resulting headings were recorded to csv files with a ROS `logger` node. Three desired closed-loop natural frequency settings were tested: $\omega_d = \{2.0, 3.0, 4.0\}$ rad/s.

These simulation results map the nonlinear frequency response H_{NL} using the tuned yaw controller of Equation (5.24) with $\kappa = 20.0$. The ratio of the amplitude of the output (actual) to input (desired) is known as the gain $|H|$ or amplitudes of the system. Amplitudes were calculated by taking half the difference of the maximum and minimum heading from respective output and input data, and the gain is shown on the Bode gain plot. The amount that the output lags behind the control input is known as the phase shift. To quantify phase shift, a peak finding function was used on the respective output and input data to store the times of local maxima into vectors. This analysis was performed for the three ω_d and displayed in Figure C.1, with the frequency presented in units of rad/s.

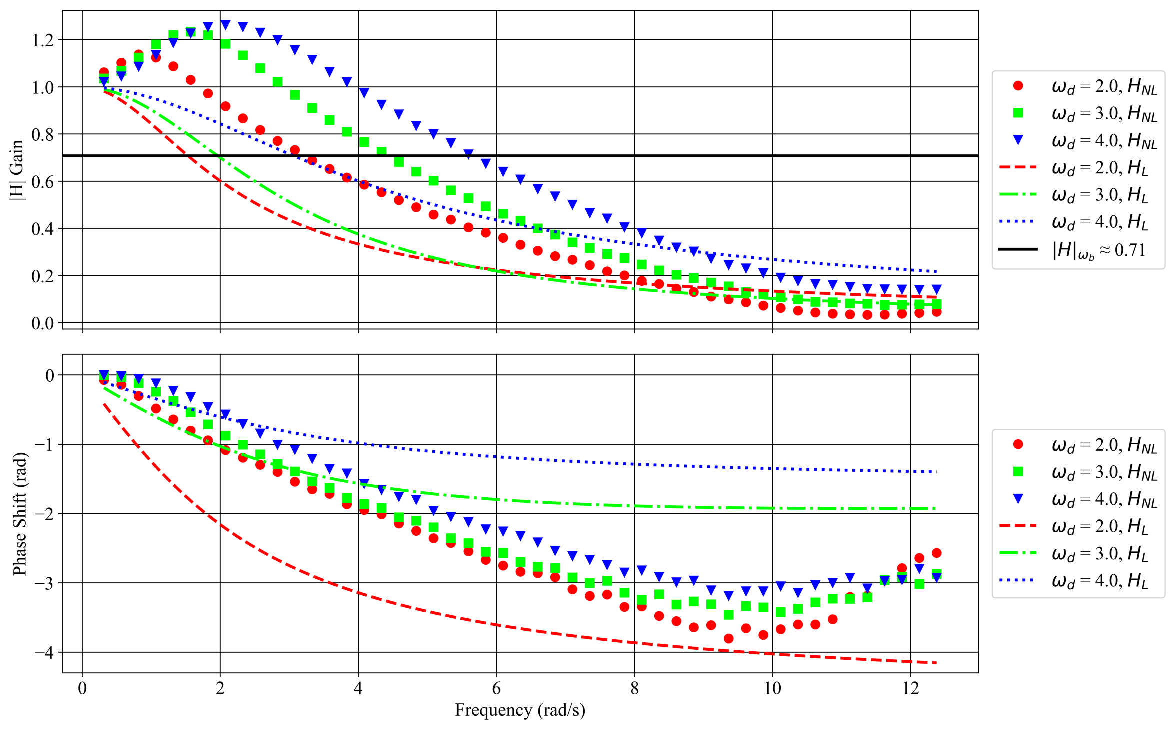


Figure C.1: Bode plot gains and phase shifts of the closed-loop nonlinear frequency response simulation results are plotted with points and responses predicted by the linear approximation are plotted with lines. A solid line representing the -3 dB control bandwidth cutoff is drawn for reference on the gain graph. At higher frequencies where phase shift exceeded $-pi$, data were unwrapped to prevent discontinuities and for clarity.

The theoretical curves for the linear transfer function H_L shown on the Bode plot were calculated using the same transfer function as Equation (5.22), solving for the frequency response by replacing $s := j\omega$, where j is an imaginary number and ω is frequency, which can be expressed as

$$|H_L| = \left| \frac{\Psi(j\omega)}{\Psi_d(j\omega)} \right| . \tag{C.1}$$

The theoretical phase shift is the angle of Equation (5.22) in the complex plane, which can be expressed as

$$\angle H_L = \arctan\left(\frac{Im(H_L)}{Re(H_L)} \right), \tag{C.2}$$

which is also a function of ω.

One common metric of assessing a closed-loop controller's response is characterizing attenuation of the frequency response with control bandwidth. The definition given by [35] states that the $-3\,\text{dB}$ cutoff is

$$|H(j\omega)|_{\omega=\omega_b} = \frac{\sqrt{2}}{2} \tag{C.3}$$

where $|H(j\omega)|$ is the gain of the frequency response of the transfer function, ω is the frequency of the desired heading, ω_n is the closed-loop natural frequency which is interchangeable with ω_d as per Equation (5.15), and ω_b is the control bandwidth which marks the highest frequency the system can keep up with before the gain falls below $-3\,\text{dB}$. In other words, the frequency ω where the gain falls to $|H| = \sqrt{2}/2 \approx 0.71$ is the so-called control bandwidth ω_b. Looking at Figure C.1, in the region spanning $0.71 \leq |H| \leq 1.0$, the slope is constant for all three tested ω_d settings, causing an apparent linear relationship. Furthermore, unity gains $|H| = 1.0$ are reached at frequencies $\omega = \omega_d$, which could serve as a custom definition of "bandwidths" for this particular nonlinear system, which could assist pilots in understanding how the choice of ω_d can be expected to facilitate BROV2 performance.

Appendix D

ROS Overview

Robot Operating System (ROS) is a middleware that provides users with many helpful features to interface with their robots. The ROS package "`mavros`" allows users to access their sensors and controls on the robots if a Flight Management Unit (FMU) using Ardupilot/ArduSub firmware. Typically these functions are accessed with a ground control program that limits user customizability. With ROS, scripts written in Python or C++ can be made to interface with the robot.

D.1 ROS Bags

The "`rosbag`" utility allows users to record all data that flows through ROS with global timestamps into a binary data type. Chronologically, the yaw portion of testing and development had used a script that recorded csv data without the use of `rosbag`. By the time depth controller testing and development was reached, proficiency in ROS had improved and `rosbag` was used. This tool was used for the gathering of simulation and real world data and was invaluable because of the global timestamps. These timestamps allowed the data analysis to be synchronized between when control signals were input and the updates recorded by the sensors with minimum hassle.

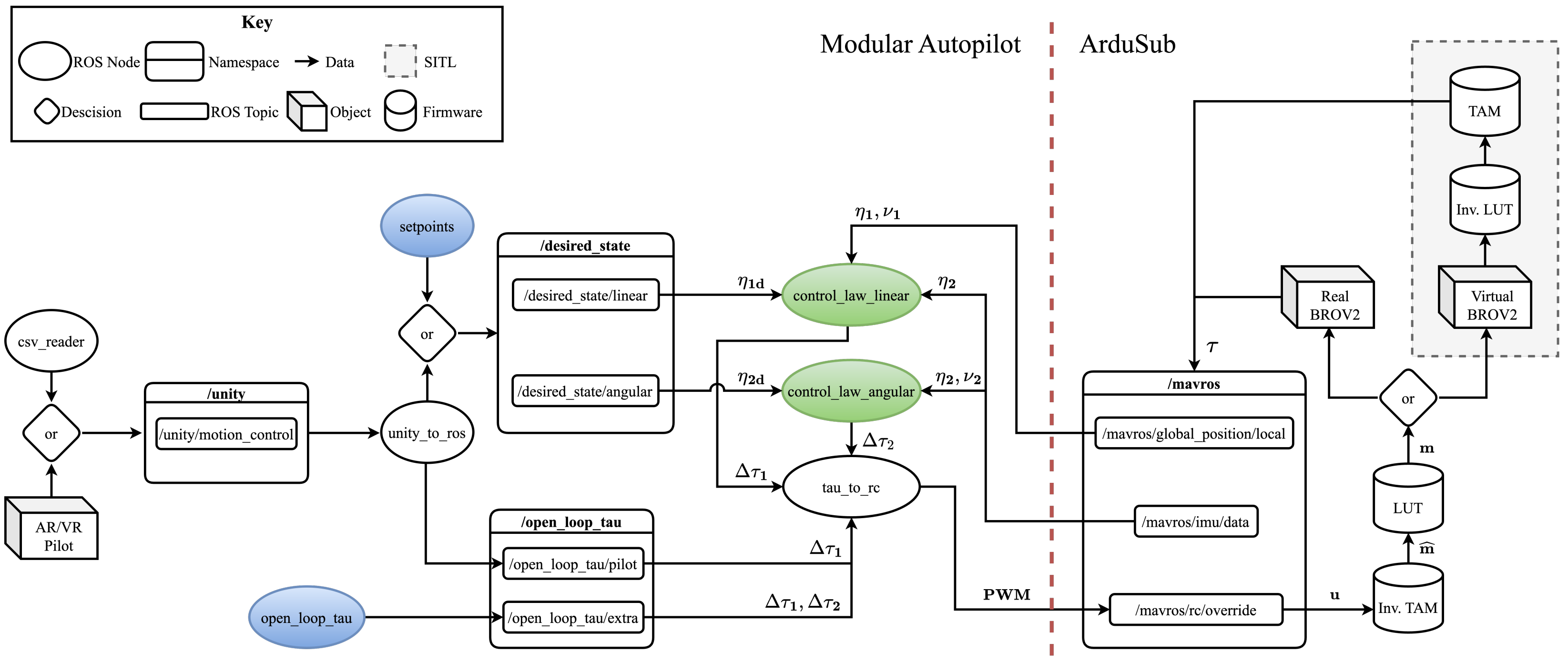


Figure D.1: A flowchart of the control architecture developed in this thesis, with custom Robot Operating System (ROS) programs shown to the left of the dashed maroon line and modifications to the ArduSub firmware shown to the right. The nodes shaded in blue represent optional nodes, those shaded in green can operate one at a time or simultaneously, and white nodes and other shapes are all part of regular operations. Firmware modifications updating the simulation models are enclosed within the gray dashed box.

D.2 ROS Nodes, ROS Topics, and Overall Control Architecture

A "`rosnode`" can generally be mapped to one script. The control architecture of the BROV2 was divided up into several nodes that were intended to handle functions in an organized and modular fashion shown in Figure D.1. The key in the upper left corner indicates what the blocks represent. The ovals are `rosnodes` labeled with the names of their Python scripts; diamonds represent decisions where only one of the two inputs can flow through. The rectangle containers with bolded headers are ROS namespaces which are used to organize `rostopics`. The rectangles inside the containers are labeled with the `rostopics` they represent. Every `rostopic` is made up of one or more `rosmsgs` which contain timestamped data vectors. The cubes represent real-world objects. The cylinders represent firmware programs or functions that were modified or added. The arrows represent the flow of data and information. The custom programs made in ROS to implement the control capabilities discussed in this thesis are shown to the left of the maroon dashed line. The modifications and additions to the ArduSub firmware discussed in Chapter 3 and Appendix B are shown to the right of the maroon dashed line.

Starting from the left: real or simulated gestures are input and mapped to the `/unity` namespace. Unity is the 3D game production program that is used as the software backend of the AR/VR setup. That data is parsed with a `unity_to_ros` node that maps the data to desired state vectors that are fed into the control laws, and open-loop linear controls are sent to the open-loop forces and torques namespace. The control laws receive sensor data and state estimations from the firmware via `mavros` which is constantly broadcasting to the `/mavros` namespace. The `tau_to_rc` node sums all input forces and torques and converts them to PWM signals that are sent to the RC override topic which generates forces and torques on the BROV2. Those forces and torques cause motion of the real or virtual vehicle while the sensors constantly update.

Nodes shaded in blue represent other possible inputs. The `setpoints` node directly sets desired coordinates and orientations. In the future these setpoints could originate from a guidance system that calculates desired states to follow waypoints in a path planning algorithm. There is a decision block to ensure control laws have only one input desired state. The `open_loop_tau` node can introduce extra forces or torques in addition to what is already commanded by the pilot and autopilot. This node would not typically be used during piloted operation and is most useful for testing purposes.

Nodes shaded in green indicate that one or both can operate. During the AR/VR pilot mode of operation both control laws would be active. The linear control law would stabilize depth and perturbations in surge and sway. The angular control law moves the vehicle in sync with the VR headset worn by the pilot. In different modes of operation the control law nodes and open-loop nodes may be inactive or command zero changes to forces or torques.

Notice the decision tree on the right of the dashed maroon line, control outputs can be routed to a virtual BROV2 which calculates the resulting body forces and torques using the hydrodynamic model or control outputs can be routed to the real BROV2. The ability to seamlessly swap between the real and virtual vehicle without changing the surrounding inputs and outputs makes SITL such a valuable tool.

D.3 ROS Standards

The contents of this appendix section directly quote [32].

In robotics applications that use the ROS, there are certain guidelines that exists known as the ROS Enhancement Proposals (REPs) [49]. Generally, the REPs are guidelines to follow that help developers integrate with the larger ROS ecosystem. They serve as valuable touchstones which provide common threads for the community to latch on to, and standardize many fundamental things that help demystify the open-source landscape. For example, the convention that all ROS vehicles follow FLU-ENU is because of REP 103 [50]. There is a longstanding issue that those using the ROS in maritime applications were urged to use FLU-ENU frames by REP 103, but the rest of the industry followed the FRD-NED, also known as forward–starboard–down, FSD-NED, frame conventions. A new REP, 156, puts this issue to rest and advises that ROS users in maritime applications preserve FLU-ENU as the primary frames, but also maintain secondary frames with the appended suffixes "`_fsd`" or "`_ned`" [51]. Readers are also encouraged to follow the ROS Maritime Working Group updates on the ROS Discourse forums [52].

D.4 Gazebo and Other Simulations

The contents of this appendix section directly quote [32].

Other simulation environments using the ROS exist, like Project Dave [53], Stonefish ROS [54], and ROS-MVP [55] that offer detailed three-dimensional (3D) graphical environments capable of simulating sensors and collision. None of these options integrate directly with off-the-shelf controllers like the SITL capabilities that are included with ArduSub, which is its major advantage. The compilation process that builds ArduSub vehicle firmwares is tied to SITL so that alterations to software that work in the virtual environment will also work on the physical vehicle. This is why simulation was done in the graphically-limited ArduSub SITL.

An alternative to the experimental strategy described here for estimating inertial properties is the moment of inertia calculator included with Gazebo Harmonic [56], or a similar tool often available in computer-aided design (CAD) programs. If users do not constrain inertia parameters themselves ahead of runtime, Gazebo can estimate the moment of inertia matrix of a robot while loading its 3D mesh model. The simulator does this with a type of finite element analysis and a density parameter given by the user. This method provides a convenient way for users to estimate inertia. However, this method does rely on the accuracy of the 3D models and the comprehensiveness of the configuration file containing joints, links, and density information. In other words, this type of estimation is reliable and convenient for simple homogeneous shapes, but becomes more cumbersome when the shapes get more complex and consist of multiple materials. This work has opted not to use Gazebo, but encouraged readers to explore their options.

References

[1] Sahoo, A.; Dwivedy, S.K.; Robi, P.S. Advancements in the Field of Autonomous Underwater Vehicle. *Ocean Engineering* **2019**, *181*, 145–160. `https://doi.org/10.1016/j.oceaneng.2019.04.011`.

[2] Petillot, Y.R.; Antonelli, G.; Casalino, G.; Ferreira, F. Underwater Robots: From Remotely Operated Vehicles to Intervention-Autonomous Underwater Vehicles. *IEEE Robotics & Automation Magazine* **2019**, *26*, 94–101. `https://doi.org/10.1109/MRA.2019.2908063`.

[3] Bovio, E.; Cecchi, D.; Baralli, F. Autonomous Underwater Vehicles for Scientific and Naval Operations. *Annual Reviews in Control* **2006**, *30*, 117–130. `https://doi.org/10.1016/j.arcontrol.2006.08.003`.

[4] Wynn, R.B.; Huvenne, V.A.I.; Le Bas, T.P.; Murton, B.J.; Connelly, D.P.; Bett, B.J.; Ruhl, H.A.; Morris, K.J.; Peakall, J.; Parsons, D.R.; et al. Autonomous Underwater Vehicles (AUVs): Their Past, Present and Future Contributions to the Advancement of Marine Geoscience. *Marine Geology* **2014**, *352*, 451–468. `https://doi.org/10.1016/j.margeo.2014.03.012`.

[5] Melo, J.; Matos, A. Survey on Advances on Terrain Based Navigation for Autonomous Underwater Vehicles. *Ocean Engineering* **2017**, *139*, 250–264. `https://doi.org/10.1016/j.oceaneng.2017.04.047`.

[6] Yoon, S.; Qiao, C. Cooperative Search and Survey Using Autonomous Underwater Vehicles (AUVs). *IEEE Transactions on Parallel and Distributed Systems* **2011**, *22*, 364–379. `https://doi.org/10.1109/TPDS.2010.88`.

[7] Trust, E.R.E. The remotely operated vehicle (ROV) Hercules is launched from E/V Nautilus. *Presence in the News* **2019**.

[8] Mcquire, J. ROV Jason is deployed from RV Sikuliaq during an Ocean Networks Canada Observatory expedition. *Environment, Coastal, & Offshore (ECO) Magazine Deep Dive III: Deep-Sea Exploration* **2023**.

[9] Schmidt Ocean Institute. The ROV SuBastian during sea trials. *WorkBoat* **2016**.

[10] School of Ocean and Earth Science and Technology (SOEST) at the University of Hawaii at Manoa. ROV Lu'ukai. Available online: `https://www.soest.hawaii.edu/UHMC/Luukai.php` (accessed on 2024-05-20).

[11] Carlsen, V.; Kandal, D.T.V. The Future of Norwegian Deep-Sea Mining: Impacts on the Sustainability Pillars. Master's thesis, University of Stavanger, Stavanger, Norway, 2024.

[12] Hudson, I.; Jones, D.; Wigham, D. A Review of the Uses of Work-Class ROVs for the Benefits of Science: Lessons Learned from the SERPENT Project. *Underwater Technology* **2005**, *26*, 83–88. `https://doi.org/10.3723/175605405784426637`.

[13] McLean, D.L.; Parsons, M.J.G.; Gates, A.R.; Benfield, M.C.; Bond, T.; Booth, D.J.; Bunce, M.; Fowler, A.M.; Harvey, E.S.; Macreadie, P.I.; et al. Enhancing the Scientific Value of Industry Remotely Operated Vehicles (ROVs) in Our Oceans. *Frontiers in Marine Science* **2020**, *7*. `https://doi.org/10.3389/fmars.2020.00220`.

[14] Mazzeo, A.; Aguzzi, J.; Calisti, M.; Canese, S.; Vecchi, F.; Stefanni, S.; Controzzi, M. Marine Robotics for Deep-Sea Specimen Collection: A Systematic Review of Underwater Grippers. *Sensors* **2022**, *22*, 648. `https://doi.org/10.3390/s22020648`.

[15] Sward, D.; Monk, J.; Barrett, N. A Systematic Review of Remotely Operated Vehicle Surveys for Visually Assessing Fish Assemblages. *Frontiers in Marine Science* **2019**, *6*. `https://doi.org/10.3389/fmars.2019.00134`.

[16] Garner, S.B.; Olsen, A.M.; Caillouet, R.; Campbell, M.D.; Iii, W.F.P. Estimating Reef Fish Size Distributions With a Mini Remotely Operated Vehicle-Integrated Stereo Camera System. *PLOS ONE* **2021**, *16*. `https://doi.org/10.1371/journal.pone.0247985`.

[17] Harmon, L.K.; Gleason, M. Underwater Explorers: Using Remotely Operated Vehicles (ROVs) to Engage Youth with Underwater Environments. *Children, Youth and Environments* **2009**, *19*, 125–143.

[18] Osen, O.L.; Sandvik, R.I.; Berge Trygstad, J.; Rogne, V.; Zhang, H. A Novel Low Cost ROV for Aquaculture Application. In Proceedings of the OCEANS 2017 - Anchorage, 2017, pp. 1–7.

[19] Aguirre-Castro, O.A.; Inzunza-González, E.; García-Guerrero, E.E.; Tlelo-Cuautle, E.; López-Bonilla, O.R.; Olguín-Tiznado, J.E.; Cárdenas-Valdez, J.R. Design and Construction of an ROV for Underwater Exploration. *Sensors* **2019**, *19*, 5387. `https://doi.org/10.3390/s19245387`.

[20] Kabanov, A.; Kramar, V.; Ermakov, I. Design and Modeling of an Experimental ROV with Six Degrees of Freedom. *Drones* **2021**, *5*, 113. `https://doi.org/10.3390/drones5040113`.

[21] Salem, K.M.; Rady, M.; Aly, H.; Elshimy, H. Design and Implementation of a Six-Degrees-of-Freedom Underwater Remotely Operated Vehicle. *Applied Sciences* **2023**, *13*, 6870. `https://doi.org/10.3390/app13126870`.

[22] Constine, J. Ocean Drone Startup Merger Spawns Sofar, the DJI of the Sea. Available online: `https://techcrunch.com/2019/03/27/sofar-ocean-technologies/` (accessed on 2024-05-20).

[23] Blue Robotics. BlueROV2 Heavy Configuration Retrofit Kit. Available online: `https://bluerobotics.com/store/rov/bluerov2-upgrade-kits/brov2-heavy-retrofit/` (accessed on 2024-05-20).

[24] Bell, T.W.; Nidzieko, N.J.; Siegel, D.A.; Miller, R.J.; Cavanaugh, K.C.; Nelson, N.B.; Reed, D.C.; Fedorov, D.; Moran, C.; Snyder, J.N.; et al. The Utility of Satellites and Autonomous Remote Sensing Platforms for Monitoring Offshore Aquaculture Farms: A Case Study for Canopy Forming Kelps. *Frontiers in Marine Science* **2020**, *7*. `https://doi.org/10.3389/fmars.2020.520223`.

[25] Zhao, L.; Zhou, M.; Loose, B.; Cousens, V.; Turrisi, R. Modifying an Affordable ROV for Under-Ice Sensing. In Proceedings of the OCEANS 2021: San Diego – Porto, 2021, pp. 1–5. `https://doi.org/10.23919/OCEANS44145.2021.9705886`.

[26] Xia, P.; McSweeney, K.; Wen, F.; Song, Z.; Krieg, M.; Li, S.; Yu, X.; Crippen, K.; Adams, J.; Du, E.J. Virtual Telepresence for the Future of ROV Teleoperations: Opportunities

and Challenges. In Proceedings of the SNAME Offshore Symposium. SNAME, 2022, p. D011S001R001.

[27] Xia, P.; You, H.; Ye, Y.; Du, J. ROV Teleoperation via Human Body Motion Mapping: Design and Experiment. *Computers in Industry* **2023**, *150*, 103959. https://doi.org/10.1016/j.compind.2023.103959.

[28] Goheen, K.; Jefferys, E. The Application of Alternative Modelling Techniques to ROV Dynamics. In Proceedings of the , IEEE International Conference on Robotics and Automation Proceedings, 1990, pp. 1302–1309 vol.2. https://doi.org/10.1109/ROBOT.1990.126180.

[29] Walker, K.L.; Gabl, R.; Aracri, S.; Cao, Y.; Stokes, A.A.; Kiprakis, A.; Giorgio-Serchi, F. Experimental Validation of Wave Induced Disturbances for Predictive Station Keeping of a Remotely Operated Vehicle. *IEEE Robotics and Automation Letters* **2021**, *6*, 5421–5428. https://doi.org/10.1109/LRA.2021.3075662.

[30] Eng, Y.; Lau, W.; Low, E.; Seet, G.; Chin, C. Estimation of the Hydrodynamics Coefficients of an ROV using Free Decay Pendulum Motion. *Engineering Letters* **2008**, *16*.

[31] Eidsvik, O.A.; Schjølberg, I. Determination of Hydrodynamic Parameters for Remotely Operated Vehicles. In Proceedings of the ASME 2016 35th International Conference on Ocean, Offshore and Arctic Engineering. American Society of Mechanical Engineers Digital Collection, 2016. https://doi.org/10.1115/OMAE2016-54642.

[32] Ng, P.; Krieg, M. Modifications to ArduSub That Improve BlueROV SITL Accuracy and Design of Hybrid Autopilot. *Applied Sciences* **2024**, *14*, 7453. https://doi.org/10.3390/app14177453.

[33] VectorNav. Reference Frames. Available online: https://www.vectornav.com/resources/inertial-navigation-primer/math-fundamentals/math-refframes (accessed on 2024-09-30).

[34] SNAME. Nomenclature for Treating the Motion of a Submerged Body Through a Fluid. *The Society of Naval Architects and Marine Engineers, Technical and Research Bulletin* **1950**, pp. 1–5.

[35] Fossen, T.I. *Handbook of Marine Craft Hydrodynamics and Motion Control*; John Wiley & Sons, 2021.

[36] Newman, J.N. *Marine Hydrodynamics*; The MIT Press, 2018.

[37] Blue Robotics. BlueROV2 Assembly, 2016.

[38] Wu, C.J. 6-DOF Modelling and Control of a Remotely Operated Vehicle. Master's thesis, Flinders University, Adelaide, South Australia, 2018.

[39] Trefethen, L.N.; Bau, D. *Numerical Linear Algebra*; SIAM, 2022.

[40] Blue Robotics. T200 Thruster Polyfit. Available online: `https://colab.research.google.com/drive/1CEDW9ONTJ8Aik-HVsqck8Y_EcHYLg0zK#scrollTo=yXoOCK3CvxoY` (accessed on 2024-05-20).

[41] Sandøy, S.S. System Identification and State Estimation for ROV uDrone. Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, 2016.

[42] Eidsvik, O.A. Identification of Hydrodynamic Parameters for Remotely Operated Vehicles. Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, 2015.

[43] Einarsson, E.M.; Lipenitis, A. MPC Control for the BlueROV2 - Theory and Implementation. Master's thesis, Aalborg University, Aalborg, Denmark, 2020.

[44] Wackerly, D.D.; Mendenhall, W.; Scheaffer, R.L. *Mathematical Statistics with Applications*; Vol. 7, Thomson Brooks/Cole Belmont, CA, 2008.

[45] Franklin, G.F.; Powell, J.D.; Emami-Naeini, A. *Feedback Control of Dynamic Systems*, sixth ed.; Pearson London, 2010.

[46] Blue Robotics. BlueROV2 Assembly (R3 Version). Available online: `https://bluerobotics.com/learn/bluerov2-assembly-r3-version/` (accessed on 2024-05-20).

[47] PX4. Pixhawk. Available online: `https://docs.px4.io/main/en/flight_controller/pixhawk.html` (accessed on 2024-05-20).

[48] APM:Sub Release Notes. Available online: https://github.com/ArduPilot/ardupilot/blob/master/ArduSub/ReleaseNotes.txt (accessed on 2024-08-13).

[49] Conley, K. REP Purpose and Guidelines. Available online: https://ros.org/reps/rep-0001.html (accessed on 2024-08-13).

[50] Foote, T.; Purvis, M. REP 103 Standard Units of Measure and Coordinate Conventions. Available online: https://www.ros.org/reps/rep-0103.html (accessed on 2024-05-20).

[51] Palmer, E.; Zhang, M. [REP-156] Define Coordinate Frame Conventions for Marine Robots #398. Available online: https://github.com/ros-infrastructure/rep/pull/398 (accessed on 2024-08-09).

[52] Discourse, R. Maritime Robotics. Available online: https://discourse.ros.org/c/maritime/36 (accessed on 2024-08-09).

[53] Zhang, M.M.; Choi, W.S.; Herman, J.; Davis, D.; Vogt, C.; McCarrin, M.; Vijay, Y.; Dutia, D.; Lew, W.; Peters, S.; et al. DAVE Aquatic Virtual Environment: Toward a General Underwater Robotics Simulator. In Proceedings of the 2022 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV), 2022, pp. 1–8. https://doi.org/10.1109/AUV53081.2022.9965808.

[54] Cieślak, P. Stonefish: An Advanced Open-Source Simulation Tool Designed for Marine Robotics, With a ROS Interface. In Proceedings of the OCEANS 2019 - Marseille, 2019. https://doi.org/10.1109/OCEANSE.2019.8867434.

[55] Gezer, E.C.; Zhou, M.; Zhao, L.; McConnell, W. Working Toward the Development of a Generic Marine Vehicle Framework: ROS-MVP. In Proceedings of the OCEANS 2022, Hampton Roads, 2022, pp. 1–5. https://doi.org/10.1109/OCEANS47191.2022.9977346.

[56] Singh, J.; Taddese, A.; Dutia, D. Proposal for Automatic Moments of Inertia Calculations. Available online: http://sdformat.org/tutorials?tut=auto_inertial_params_proposal (accessed on 2024-08-09).