# DESIGN AND SIMULATION OF A CONTROL CONTINUUM FOR TETHERLESS UNDERWATER VEHICLES

by

Graham LeBlanc

Submitted in partial fulfillment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
August 2011

DALHOUSIE UNIVERSITY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled “DESIGN AND SIMULATION OF A CONTROL CONTINUUM FOR TETHERLESS UNDERWATER VEHICLES” by Graham LeBlanc in partial fulfillment of the requirements for the degree of Master of Applied Science.

Dated: August 24, 2011

Supervisors: ______________________________

______________________________

Readers: ______________________________

______________________________

DALHOUSIE UNIVERSITY

DATE: August 24, 2011

| | |
|---|---|
| AUTHOR: | Graham LeBlanc |
| TITLE: | DESIGN AND SIMULATION OF A CONTROL CONTINUUM FOR TETHERLESS UNDERWATER VEHICLES |

DEPARTMENT OR SCHOOL: Department of Electrical and Computer Engineering

DEGREE: M.A.Sc. CONVOCATION: October YEAR: 2011

Signature of Author

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ABSTRACT

There exists a need for a new class of underwater vehicle that can perform both close control tasks, as well as long-range exploration, without manual reconfiguration. A tetherless underwater vehicle (TUV) with acoustic communications to an operator station has potential to fulfill this need, while also removing much of the operating costs associated with tether management. The problem with TUVs is the limited communications bandwidth and time lag increasing with range from the transmitter. This thesis introduces a new class of controller for TUV vehicles that isolates the operator from the time-varying delay. This isolation is achieved through the formation of a continuum of control comprised of existing control paradigms, such as predictive and autonomous control. A smooth evolution through the continuum is formulated based on the time delay. The resulting controller permits operator close control for extended ranges without manual reconfiguration of the vehicle or controller.

# List of Abbreviations and Symbols Used

| Symbol | Definition |
|---|---|
| AUV | Autonomous underwater vehicle |
| DOF | Degree(s) of freedom |
| DVL | Doppler velocity log |
| FPS | Frames per second |
| INS | Inertial navigation system |
| MPC | Model predictive controller |
| MRDS | Microsoft Robotics Developer Studio® |
| PID | Proportional-integral-derivative (controller) |
| RK4 | Runge-Kutta 4th-order method |
| ROV | Remotely operated vehicle |
| SLAM | Simultaneous localization and mapping |
| SMC | Sliding mode controller |
| TUV | Tetherless underwater vehicle |
| USBL | Ultra-short baseline (positioning system) |
| UUV | Unmanned underwater vehicle |
| $Y_h(s)$ | Human operator transfer function |
| $K_h(s)$ | Human operator control gain |
| $\tau_e$ | Effective time delay of human operator |
| $T_L$ | Human operator lead time constant |
| $Y_c(s)$ | Inertial system transfer function |
| $K_c$ | System gain |
| $\tau$ | System time constant |
| $w_c$ | Open-loop cross-over frequency |
| $T_D$ | Pure delay control loop element |
| $\tau_d$ | Delay element time constant |
| $\tau_e$ | Total loop time delay time constant |

| Symbol | Definition |
|---|---|
| $h$ | Prediction or Euler time span |
| $\mathcal{L}\{\cdot\}$ | Laplace transform operator |
| $x$–$y$–$z$ | Body-fixed reference frame |
| $X$–$Y$–$Z$ | Inertial (Earth-fixed) reference frame |
| $Y(\psi)$–$Z'(\theta)$–$X''(\phi)$ | Intrinsic rotations for vehicle orientation |
| $\psi, \theta, \phi$ | Yaw, pitch, roll Euler angles |
| $\boldsymbol{\eta}$ | Vector containing inertial frame position and orientation of vehicle |
| $\boldsymbol{r}, \boldsymbol{\beta}$ | Vehicle position and orientation vectors |
| $\dot{\boldsymbol{\zeta}}$ | Vector containing body-fixed translational and rotational velocities |
| $\mathbf{T}_\beta, \mathbf{T}_1, \mathbf{T}_2$ | Rotation transformation matrices |
| $\mathbf{M}, \mathbf{M}_A$ | Vehicle mass and added mass matrices |
| $\boldsymbol{F}_B$ | Vehicle buoyancy force |
| $\boldsymbol{F}_D$ | Vehicle hydrodynamic drag force |
| $\boldsymbol{F}_C$ | Vehicle Coriolis force |
| $\boldsymbol{u}(t)$ | Vehicle external forces |
| $\boldsymbol{F}_T$ | External thruster force |
| $\boldsymbol{\epsilon}$ | Disturbance force |
| $\dot{\boldsymbol{\zeta}}_{\text{rel}}$ | Vehicle’s velocity vector relative to water |
| $\boldsymbol{v}_c$ | Ocean current velocity |
| $\mathbf{I}$ | Vehicle inertia tensor |
| $\mathbf{r}_B$ | Centre of buoyancy in body-fixed frame |
| $k_\zeta, k_{\zeta\|\zeta\|}$ | Linear and quadratic drag coefficients |
| $T_{\text{max}}$ | Maximum (saturation) thruster force |
| $\boldsymbol{\Gamma}_T$ | Mapping of thruster forces to body-fixed frame |
| $t_{c/p}, t_{p/s}, t_{s/a}$ | Key time delays in control continuum |
| $T_{wp}$ | Waypoint creation period |
| $T_{\text{meas}}$ | Actual vehicle state measurement period, or ghost correction period |
| $\boldsymbol{T}_{\text{op}}$ | Operator commanded thruster forces |

| Symbol | Definition |
|---|---|
| $\boldsymbol{T}_{\text{PID}}$ | Autopilot commanded thruster forces |
| $\boldsymbol{T}$ | Auto-op mixer commanded thruster forces |
| $\boldsymbol{x}, \boldsymbol{x}_d, \boldsymbol{x}^*$ | Measured, desired, and predicted/waypoint vehicle state |
| $\boldsymbol{\varepsilon}_{\text{PID}}$ | PID state error vector |
| $\|\boldsymbol{\varepsilon_c}\|, \varepsilon_c$ | Ghost vehicle correction distance |
| $\|\boldsymbol{\varepsilon_p}\|, \varepsilon_p$ | Prediction error distance |
| $\varepsilon_c^{\text{max}}$ | Ghost correction distance threshold |
| $K_\varepsilon$ | Auto-op command mixing factor |
| $\varepsilon_p^{\text{low}}, \varepsilon_p^{\text{high}}$ | Auto-op mixing parameters |
| $e_s$ | Actuator error (difference between actual and commanded output) |
| $\Delta\psi, \Delta\dot{\psi}$ | PID heading errors |
| $\Delta r_x, \int \Delta r_x dt, \Delta\dot{r}_x$ | PID surge DOF errors |
| $\Delta r_z, \int \Delta r_z dt, \Delta\dot{r}_z$ | PID sway DOF errors |
| $\Delta r_y, \int \Delta r_y dt, \Delta\dot{r}_y$ | PID depth errors |
| $K_\psi^P, K_\psi^D$ | PID heading gains |
| $K_{x/y/z}^P, K_{x/y/z}^I, K_{x/y/z}^D$ | PID translational gains |
| $K_T$ | PID anti-windup compensator gain |
| $q, q_0, q_{\text{rot}}$ | Simulator rotation quaternions |

# ACKNOWLEDGEMENTS

First and foremost I thank my thesis co-supervisors Dr. Peter Gregson and Dr. Jason Gu. Their knowledge and expertise gave me the inspiration for this thesis and provided me with the support needed to see it through to completion.

A very special thanks also goes to Dr. Mae Seto and Mr. Mark Rowsome of DRDC Atlantic. Their extensive knowledge and experience in the field assisted greatly in this research. With their help, I was introduced to actual ROV vehicles and was able to pilot one. They also assisted me with ensuring my research remained practical and useful in a real-world environment. Additionally, I would like to thank Dr. Jacek Ilow for sitting on my examining committee and supervising this thesis.

I am also most grateful for the unrivaled support from my parents, Janet and Don, and the rest of my family and friends, especially my girlfriend, Courtney. Their love and understanding guided me through this process and encouraged me to deliver my best work.

This research has been funded by an NSERC Alexander Graham Bell Graduate Scholarship, an APENS Engineering Centennial Scholarship, and Dalhousie in-course graduate scholarships. My deepest gratitude is extended to these institutions for their financial support.

# CHAPTER 1

# INTRODUCTION

Most of the Earth's surface is covered by ocean and yet we still do not have a full understanding of this predominant and important environment. To aid in the exploration of the vast ocean habitats, humans began employing underwater vehicles as early as 1775 with a submersible called "Turtle" [1]. This was a simple, wooden submarine that held one occupant, but began a new era of underwater exploration and discovery.

As the depths became greater and the danger to human operators grew, unmanned underwater vehicles (UUVs) were used in place of manned submarines. Today, UUVs play a pivotal role in the academic, military and commercial underwater sectors. One such example is with the recent *Deepwater Horizon* oil spill that began April 20, 2010. The company in charge, BP, employed a fleet of underwater vehicles to aid in the containment of the spill [2]. These advanced, remotely-controlled, robots were assigned tasks as meticulous as removing six bolts from a wellhead, at depths exceeding 1.5 km underwater. This task required precise control of manipulators attached to the robots, as well as fine position control to keep the vehicle platform stable. Without the aid of the robots, the task would be almost impossible due to the harsh environment that was too dangerous for humans to directly deal with.

With the increasing complexity, and danger of underwater operations, the demand for advanced underwater robots has increased significantly [3]. With current systems, there is no easy way to switch between different operations. For example, switching between close to exploratory missions may require a different control strategy, vehicle configuration, or even a different vehicle system altogether. The research presented here presents a new class of underwater robot controller that can accommodate a wide range of tasks and operating distances with no manual reconfiguration.

## 1.1 State of the Art in Unmanned Underwater Vehicles

The last decade has shown a significant increase in interest in the development and application of UUVs. It is due to this increase that UUV technology is moving from a purely academic and research field, to large, commercial markets [1]. With the commercial backing, there has been a considerable amount of recent research and development in the UUV sector. In the past, many UUV systems were made specifically with one application in mind and that is all they were able to perform. Recently, research has focused on making vehicle platforms that can accomplish many different tasks. This thesis expands on the current trend of generalization of underwater robots.

### *1.1.1 Existing UUV Systems*

With the abundance of underwater developments, there are far too many UUV systems currently available to list all of them here. Instead, this section attempts to present a cross-section of current technologies and capabilities. The goal is to give the reader an idea of the variety of vehicles that currently exist and the tasks they are designed to accomplish. It is useful to analyze each type of UUV system to develop a full understanding of the limitations imposed, depending on the technology employed.

#### Remotely Operated Vehicles

Remotely operated vehicles (ROVs) are one of the more popular types of submersible system. This is due, in part, by the fact they are connected to the surface through a tether. The tether not only allows an operator to remain in constant communication with the vehicle, it also allows power to be delivered from an external source. The constant communication greatly reduces vehicle complexity and the external power source reduces the weight of the vehicle and can allow for longer operational times. Figure 1.1 shows a typical ROV deployment scenario with a surface vessel being required for tether and vehicle management. The vehicle operator is usually stationed on the surface vessel and sends/receives telemetry to/from the ROV through the tether and is presented with instantaneous feedback from the vehicle. The feedback is routed to an operator station where it displays video from on-board cameras and various sensor readings, as shown in Figure 1.2. In some special cases, such as the work outlined by Bulich *et al.* [39], a time delay may be inevitable. In this work, the ROV is controlled through an internet connection that results time delays in the order of several seconds.

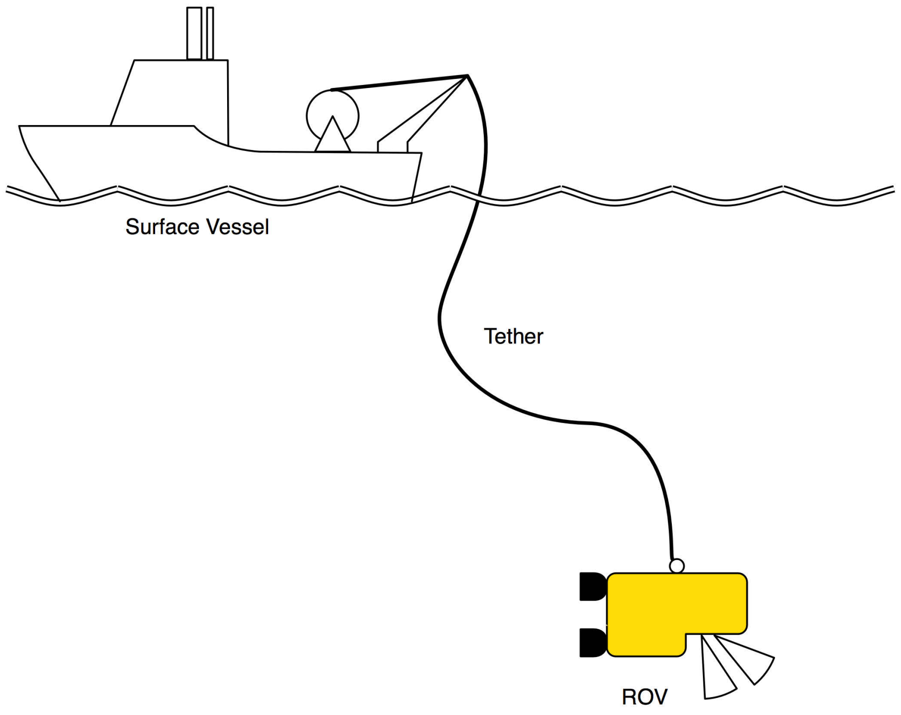


**Figure 1.1:** A typical ROV deployment with a surface vessel for management of the tether connected to the underwater vehicle.

One of the main attractors of ROVs is their versatility and the large variety of configurations available. ROVs can be very small such as the Seamor line of ROVs [4] which weigh only 20 kg, to large vehicles such as the ROPOS ROV [5] weighing in at 2630 kg (see Figure 1.3). Although these are but two examples they illustrate the wide array of ROV systems. Another advantage of ROVs is that the operator remains in the control loop with no delay between when a command is issued and when it is received by the vehicle. This allows for unencumbered, instantaneous control by the operator and exploits a human's ability to adapt to unforeseen events.

The technologies that enable these advantages also create some major drawbacks. First, the tether length imposes a restriction on the operating range of the vehicle. Second, the surface vessel and the required crew incur high costs throughout the duration of the

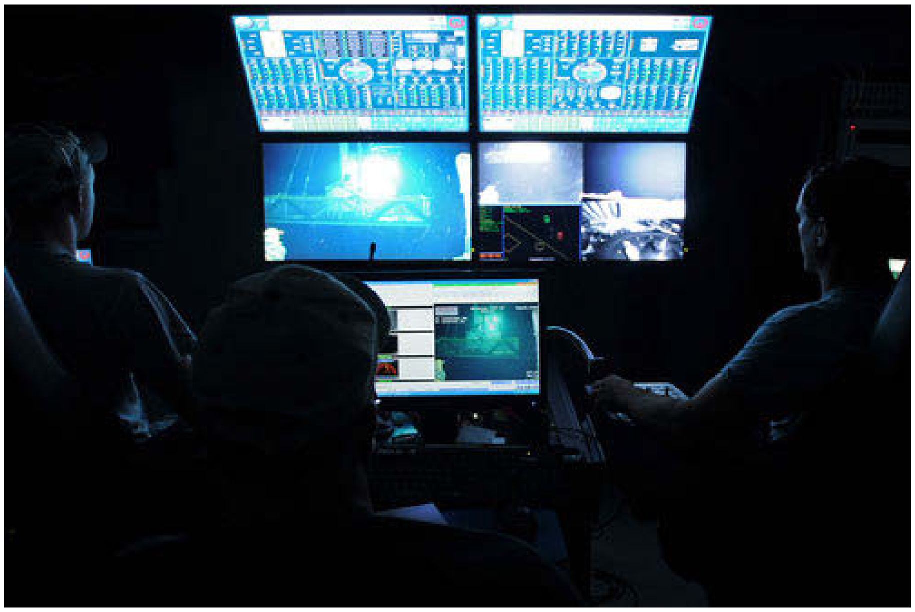


**Figure 1.2:** ROV operator station as used by BP during the Deep Water Horizon Oil Spill clean-up (*with permission*: `http://cgvi.uscg.mil/media/main.php?g2_itemId=863507`).

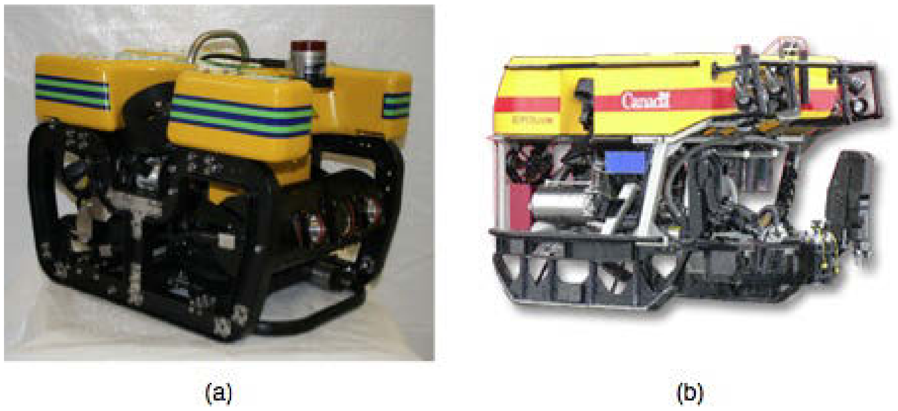


**Figure 1.3:** Images of two existing ROV systems. The smaller vehicle (a) is the Seamor 300T weighing only 20 kg (*with permission*: `http://www.seamor.com/2009/05/seamor-300t.html`), while the larger vehicle (b) is the ROPOS ROV weighing in at 2630 kg (*with permission*: `http://www.ropos.com/index.php?option=com_content&view=article&id=16&Itemid=20`).

operation. The ROV crew alone can cost $10,000 per day or more [6] with the vessel and its crew costing an additional $10,000 to $100,000 per day (M. Seto, DRDC Atlantic, private communication. July 2011). Apart from daily costs, there is also the risk and cost associated with tether entanglement on underwater debris. In some circumstances it may be required to deploy additional vehicles or dive crews to free the vehicle. In the worst case, it is possible the vehicle may need to be abandoned completely. Approximately 10% of ROVs are lost in this manner [3].

**Autonomous Underwater Vehicles**

The other type of popular underwater robot is the autonomous underwater vehicle (AUV). Unlike ROVs, AUVs do not have a tether connected to the surface. The absence of a tether means the vehicle must store its own energy on-board. The main advantage of this is that it removes all costs associated with the tether and eliminates the risk of tether entanglement. Although there is no tether, some AUVs are still capable of communicating with an operator station (although not continuously like ROVs). These types of AUVs commonly use an underwater acoustic link (as in [7–9]) or surface link for radio communications (as in [3, 8]). An illustration of the two typical AUV communication methods is shown in Figure 1.4. Many AUVs also take advantage of the more energy efficient torpedo-like configuration such as the Durado AUV (shown in Figure 1.5), unlike ROVs which often use open-frames. The efficient design puts less strain on the on-board power source, allowing for a greater operating range (some AUVs can operate in excess of 2000 km on a single charge [10]).

While the advantages allow AUVs to travel greater distances and have lower costs, the removal of the tether also removes the human operator from the control loop. This reduces the complexity of the tasks the AUV can perform. The problem is compounded by the fact that torpedo-like configurations usually provide less maneuverability than their open-frame counterparts. For this reason, much recent research is focused on a new type of AUV system called a hovering AUV [11–15]. This system borrows the open-frame design like many ROVs while keeping the autonomy and energy efficiency of AUVs. They are referred to as “hovering” vehicles since they have a sufficient number of thrusters to control each DOF and can effectively “hover” or hold station (like many ROVs). A popular application for these vehicles is for ship hull or seafloor inspection tasks. Both Vaganay *et al.* [15] and Beaujean *et al.* [11] propose systems that utilize either a tether or acoustic

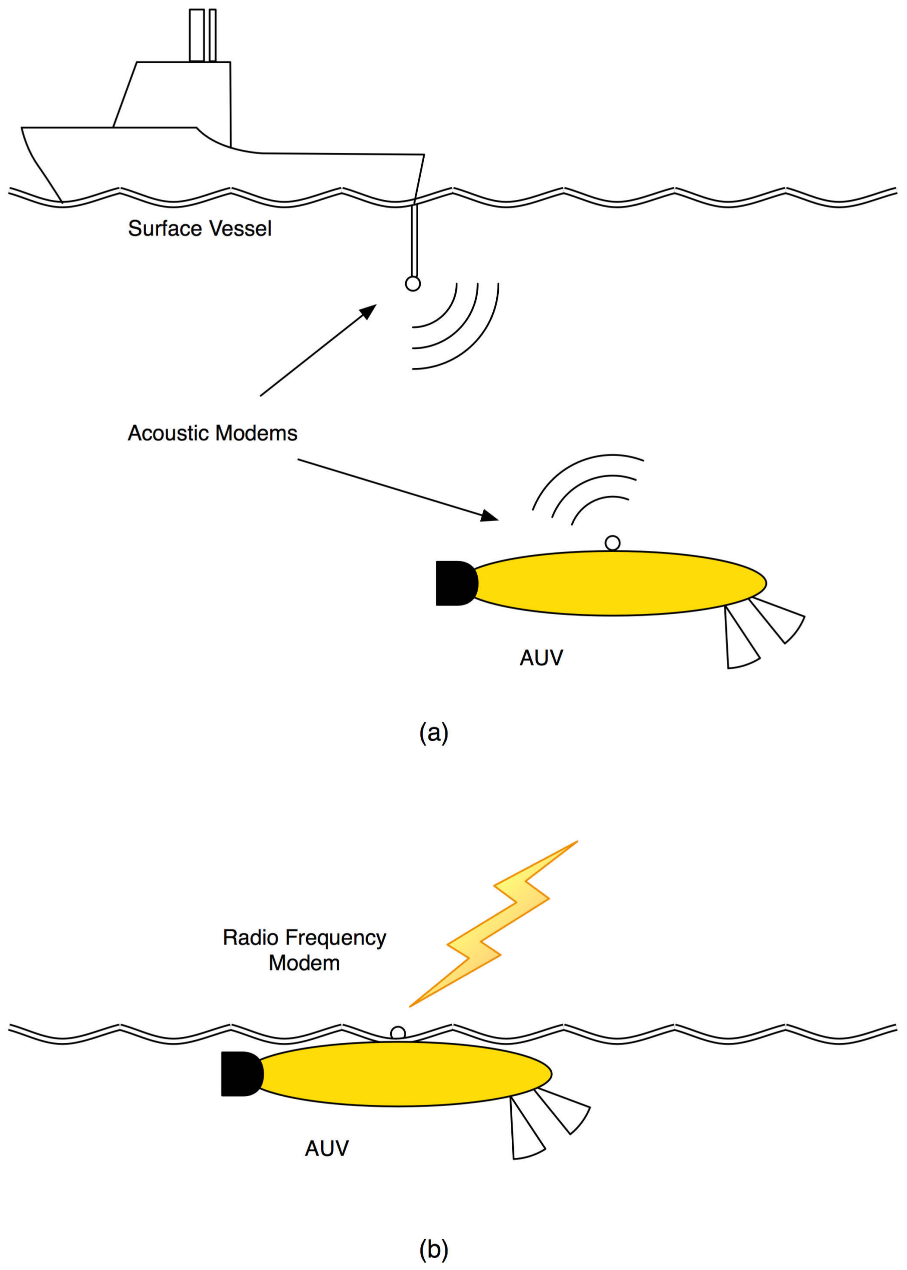


**Figure 1.4:** Two typical AUV configurations: (a) the AUV communicates periodically with the surface vessel through an acoustic link; (b) the AUV needs to surface in order to establish a radio frequency communications link.

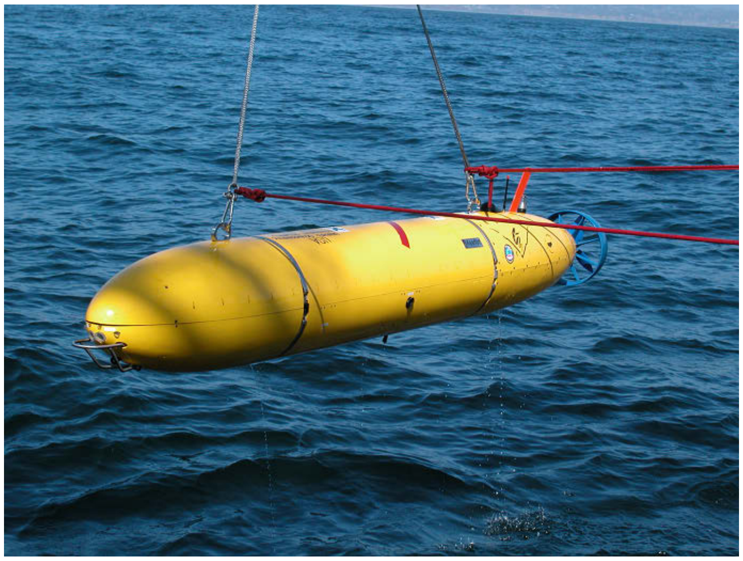


**Figure 1.5:** The Durado AUV, a torpedo-like AUV configuration (*with permission*: `http://www.mbari.org/aosn/AOSN_MB2003table.htm`).

link for high resolution image transfer of a ship's hull. Both of these systems also allow a certain degree of operator control in additional to autonomous control of the vehicle; however, this manual control is limited to short ranges up to a maximum of 130 m [11].

### Tetherless Underwater Vehicles

One other class of UUV system exists that is less widely used than the previous two. This class of vehicle is referred to as tetherless underwater vehicles (TUVs) (note, some works such as [1, 16] refer to them as unmanned untethered vehicles, not to be confused with unmanned underwater vehicles). TUVs try to combine the salient features of both ROVs and AUVs into one underwater vehicle package. Specifically, TUVs have no physical connection to the surface, yet remain in constant, real-time communication with a surface operator station. Often, this is accomplished through an acoustic link from the vehicle to the operator station. The acoustic link is used to send control data from the operator to the vehicle and both video and state telemetry from the vehicle back to the operator station. Like ROVs, a human operator can be in constant control of the vehicle, allowing it to perform tasks of higher complexity than those of an AUV, but without the burden of a tether like ROVs. The main advantage of a TUV system is the greatly-reduced operating costs with the removal of the tether and possibly the surface vessel as well. Like AUVs, this could reduce costs anywhere from $10,000 to $100,000 per day, over ROV systems.

Various configurations are possible to ensure the acoustic link maintains constant communication with the vehicle. One set of configurations uses only a small surface vessel which could be either manned or autonomous. The surface vessel tows an acoustic transceiver under the water at a shallow depth, as depicted in Figure 1.6(a). The transceiver then communicates with the TUV at a range that is limited by the chosen acoustic modems (more on this in §1.1.2). Ballou [16] used this method for his application of a TUV for underwater maintenance tasks. For this application, the vehicle was required to remain in constant communication in ranges up to 1 km (an analysis of this application is done in §1.2).

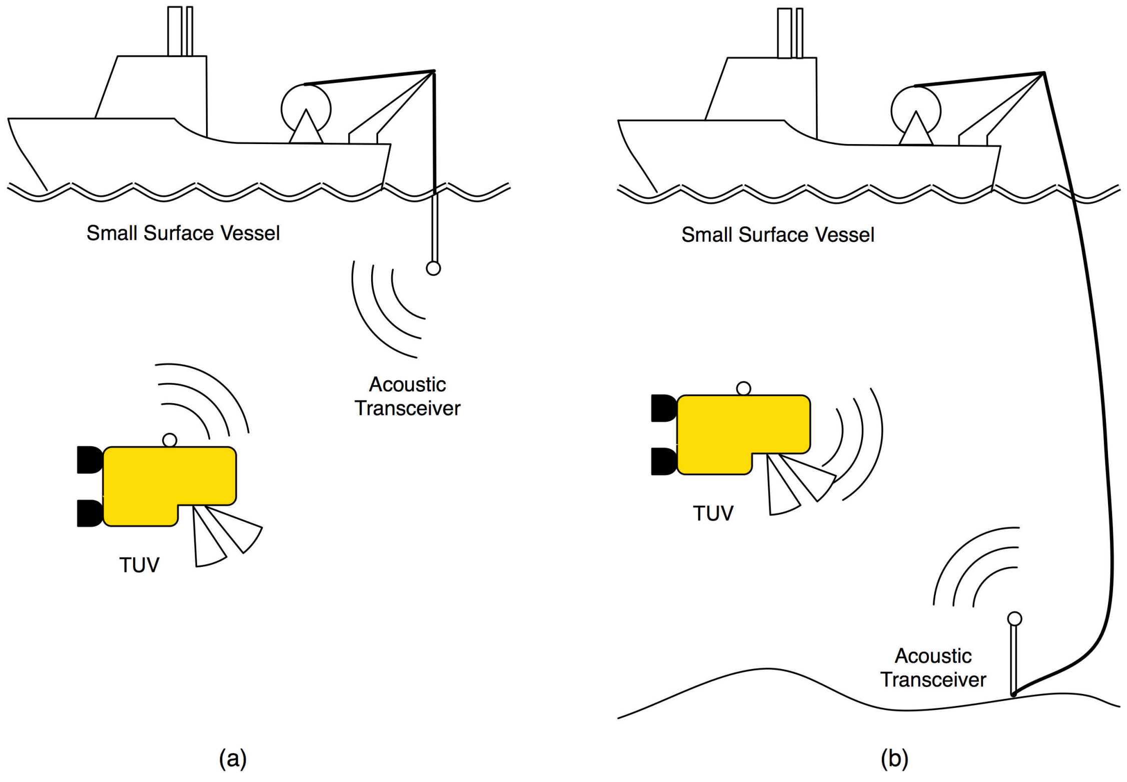


**Figure 1.6:** Two configurations of a TUV communication and control system using a small surface vessel: (a) the surface vessel tows an acoustic transceiver; (b) the surface vessel is connected to an underwater transceiver through a cable.

Another method, still using a small surface vessel, removes the transceiver from the surface vessel and instead places it on the ocean floor. The surface vessel is then connected to the bottom transceiver with a cable, as shown in Figure 1.6(b). The bottom transceiver is advantageous since it both reduces the acoustic distance to the TUV, increasing the reliability of the communications channel, while also increasing the operating range

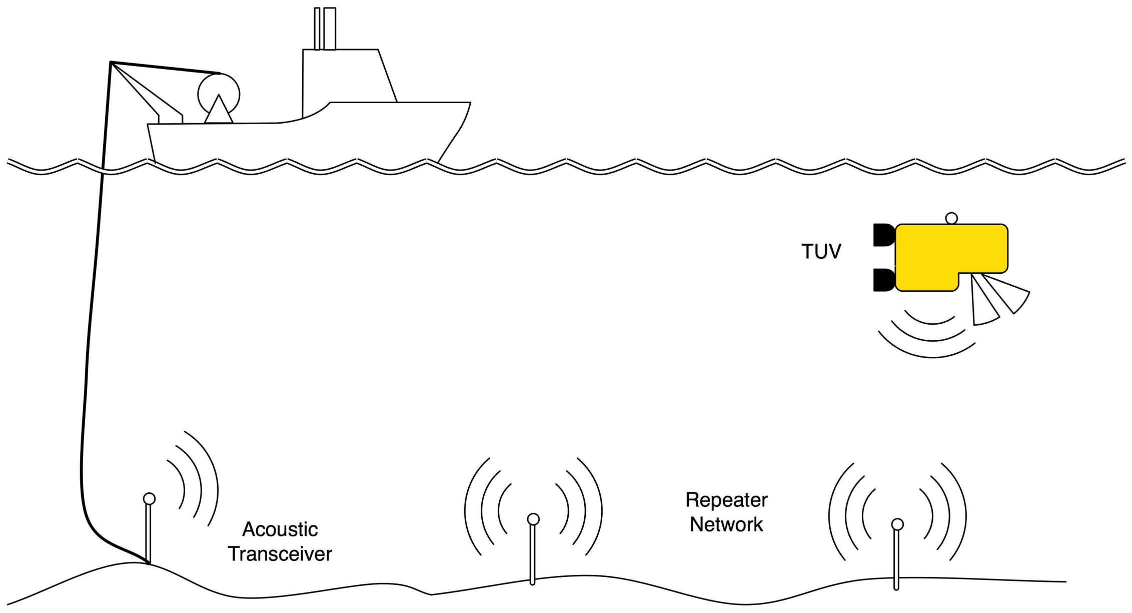


**Figure 1.7:** An acoustic communications network comprised of acoustic repeaters for facilitating long-range communication with an underwater vehicle.

correspondingly with the length of cable deployed. It is also possible to further increase the operating range by using any number of acoustic repeaters, forming an underwater acoustic network [7, 8, 17, 18]. These networks have been used successfully to facilitate communication among multiple AUVs and can carry data reliably for many kilometres, as illustrated in Figure 1.7.

One variation of the above mentioned configurations would be to remove the surface vessel completely. Instead, the vessel would be replaced with a surface gateway buoy [7, 17], as shown in Figure 1.8. This can almost completely remove the daily operating costs apart from the operators themselves. In this case, the vehicle and buoy could simply be dropped off at the area of interest and operated remotely at a different location. With all of the variations and robustness, TUVs have great potential to become a strong contender in the underwater robot industry. It may also be possible to adapt an existing hovering AUV system to fulfill many of the same needs and effectively transform it into a TUV.

### *1.1.2 Acoustic Communication Systems*

One of the most important aspects of UUV systems is a reliable communications link. ROVs have this problem solved with the physical tether to the surface. The link can be either electrical or optical, allowing for extremely high-bandwidth connections to an

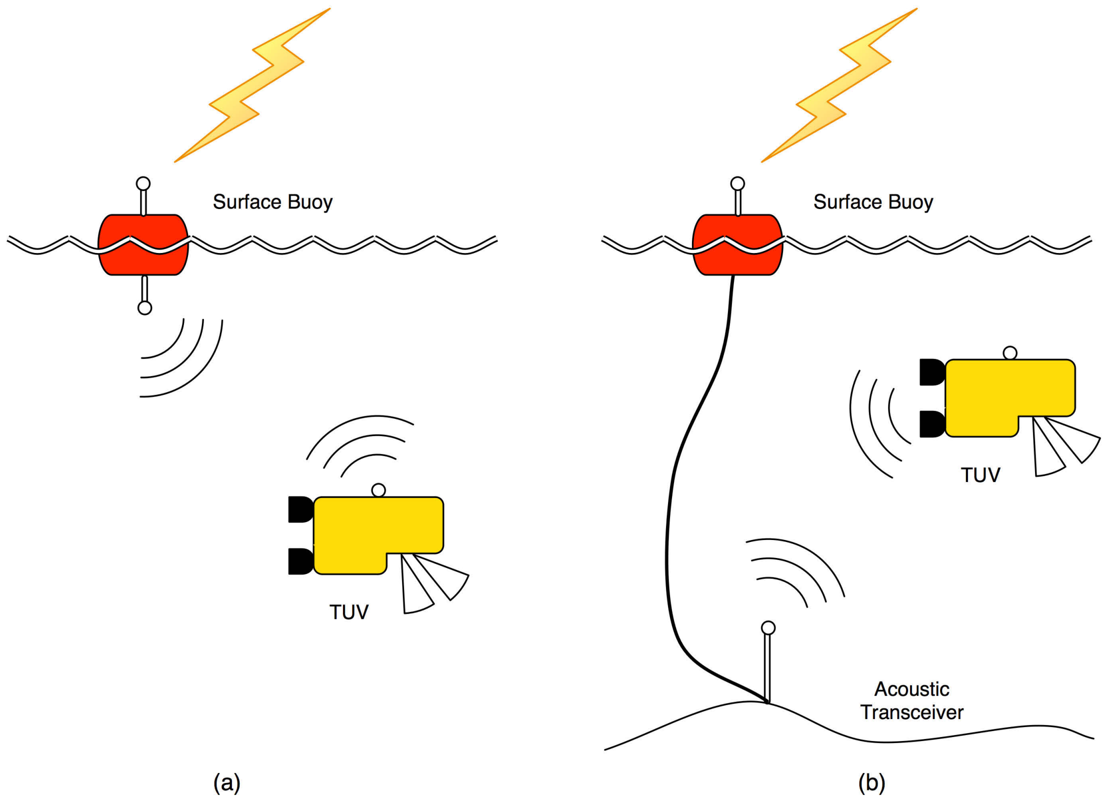


**Figure 1.8:** Two configurations of a TUV communication and control system using only a buoy with an RF link to the operator station, instead of a surface vessel: (a) the buoy uses an acoustic modem to communicate with the TUV; (b) the buoy is anchored to an underwater acoustic transceiver.

operator. When the tether is removed, we must rely on creating a wireless link in the harsh ocean environment. Radio waves will only propagate through the conductive ocean water at low frequencies (30–300 Hz) and require costly receivers and transmitters [17]. This is not feasible for UUVs since they could not handle the power or weight requirements for such a system. Blue-green lasers do not suffer as much from attenuation compared to RF, but are instead afflicted by the optical scattering effects of the water column. A UUV equipped with an optical link would need to achieve nearly impossible accuracy in aiming the laser on the moving UUV platform.

Although an active area of research, acoustic transmission is still the most reliable and cost-effective means to achieve wireless, underwater communications [16, 17]. The unique drawback of an acoustic link is the introduction of a time-varying, communication time delay. This delay arises due to the propagation speed of sound through water (approximately 1500 m/s). One can expect a round-trip time delay of 0.5 s at only 400 m and since a UUV is a moving platform, the delay will vary as the vehicle's range from the transceiver varies. The implication to the control system from this delay is discussed in §1.1.4, with a more complete discussion of control through time delay in §2.1. Since it is not the focus of this thesis, a thorough investigation of acoustic communications will not be presented here. Rather, a snapshot of current research and capabilities will be described in order to justify the application in this investigation.

For the results of this research to be practical, the acoustic link must facilitate the transfer of control data telemetry to/from the TUV and real-time video from the vehicle to operator station. Control data only requires a bandwidth of 1 kbps, whereas real-time video can require in excess of 10 kbps [17]. Since an underwater acoustic channel has only limited bandwidth, it is important to use the small amount available as efficiently as possible. This can be accomplished by improving both the acoustic link and video compression techniques. Firstly, the video need only be of a quality sufficient for a human operator to navigate the vehicle. Given the low-contrast environment, and usually well-defined objects of interest, the bit depth can then be set to 2 bits per pixel [16, 17]. Combined with current compression techniques resulting in a 250:1 compression ratio [18], 256x256 video can be sent at 15 frames per second at less than 10 kbps. In fact, real-time video has been sent through an underwater acoustic link as early as 1989 by Kaya and Yauchi [19]. This application was only short range (60 m), but demonstrated that it is possible and an active area of research. Kaya and Yauchi's method also exploited the fact that video telemetry requires the most bandwidth and is only one-way, thus they implemented an asynchronous scheme with a higher upload rate (from vehicle to station) than download. Another asynchronous method, that illustrates the use of an acoustic link for image data, is that implemented by Beaujean and Carlson [11] for their hovering AUV. Their method demonstrated data rates up to almost 90 kbps in challenging, shallow environments.

According to Stojanovic [17], the bandwidth-range product of an underwater acoustic environment is approximately 100 km*kHz. With present systems achieving a bandwidth efficiency of $\sim$1 bps/Hz, data rate-range products can be realized in the order of 100 km*kbps [17, 18]. With the above requirement of 10 kbps, current systems should be able to handle real-time video out to a range of 10 km. This range is expected to increase even further with current research suggesting bandwidth efficiencies exceeding 4 bps/Hz and video compression ratios up to 1500:1 [16]. It has already been demonstrated that data telemetry can be sent at horizontal ranges up to 200 km [20]. With advances in video transmission, real-time control of a remote vehicle through an acoustic link will be possible for very longe-range missions.

### *1.1.3 Navigation Systems*

Since GPS does not penetrate below the surface, it is difficult to maintain an accurate measurement of the vehicle's position. Early AUVs used only dead reckoning for their pose estimates [1]. It is desirable to employ a more accurate positioning method than dead reckoning alone, so modern UUVs use an array of sensors and amalgamate their readings into one pose (or state) estimate. The main type of navigation system used today is the ultra-short baseline (USBL) acoustic positioning system. Here, the vehicle is positioned by measuring the travel time of acoustic pulses between the surface vessel and the UUV (as illustrated in Figure 1.9). While an inexpensive and easy system to configure, the measurements are only reported at an accuracy of $\pm$15 m. This is largely due to calibration errors, ship motion and measurement noise. While this accuracy may be sufficient for simple tasks, complex ones like work site inspection, wreck investigation, and map building require higher degrees of accuracy.

The USBL system can be augmented by the addition of supplementary sensor readings. Both Steinke [21] and Leader [22] discuss a method that uses a Kalman filter to blend the

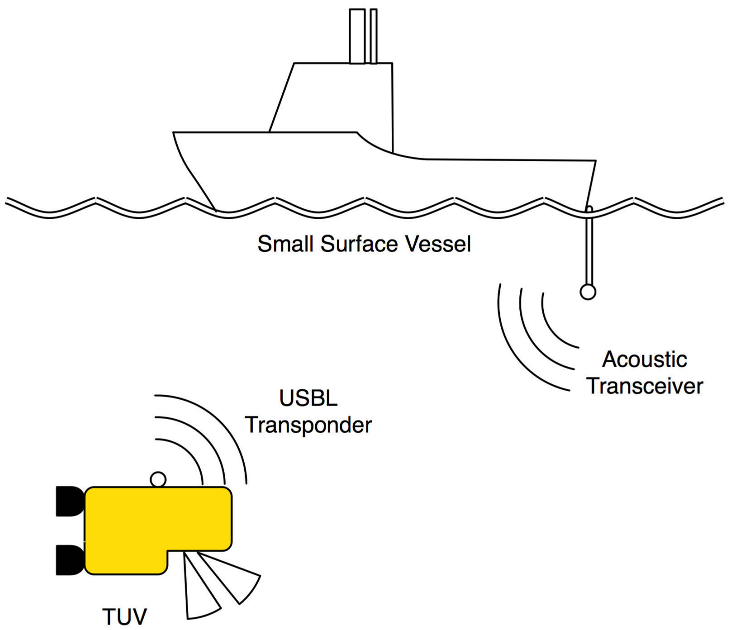


**Figure 1.9:** Illustration of a USBL navigation system. The surface vessel tows a transceiver that sends acoustic pings and waits for the response from the transponder on the remote vehicle.

measurements from multiple sensor packages. In addition to USBL, a Doppler velocity log (DVL) is used to measure the vehicle-relative velocity. DVLs are another acoustic sensor that measure the Doppler effect of reflected acoustic beams off the sea floor. For the vehicle's attitude and heading, an inertial navigation system (INS) is used. Specifically, Steinke uses a gyrocompass which also incorporates accelerometers to further augment the USBL's position estimation. With gyrocompasses, heading is relative to true north since they measure the Earth's rotation. The problem with INSs is since they integrate the readings over time, error accumulates causing the readings to drift. The Kalman filter method corrects for this by using other sensor measurements to determine if the INS unit's measurements are drifting and puts less importance on them if so. Finally, a very accurate depth measurement is obtained using a Paroscientific pressure sensor. Combining these measurements, Steinke was able to achieve a positioning accuracy of $\pm$2.5 m at ranges of 2500 m. While an improvement over USBL alone, a 2.5-m error is still large when tasks require precise motions and positioning.

A further increase in accuracy can be observed when introducing computer vision-based techniques for navigation. Myagotin and Burdinsky [23], use acoustic positioning combined with computer vision to locate known underwater targets with visual pattern recognition. This system is able to achieve a lateral accuracy in the range of centimetres, and orientation within 5°. The approach of Beckman *et al.* [24] is slightly different in that they use the terrain below the vehicle to locate itself on bathymetric maps (referred to as terrain matching). Again, accuracy with this method is also in the range of centimetres. Lastly, Dalgleish *et al.* [25] create an artificial visual target by projecting a laser stripe onto the ocean floor. From this, computer vision can be used to extract the profile of the area below the vehicle. If bathymetric data does not exist for a certain area, then techniques like this could be used to perform simultaneous localization and mapping (SLAM), as is used by Vaganay *et al.* [15]. Their approach combines both visual and acoustic imaging with existing navigational data to localize the vehicle relative to a ship's hull or the seafloor.

### *1.1.4 Control Paradigms*

Another crucial component of a UUV package is the command and control subsystem. Underwater vehicles are difficult to control due to the highly-nonlinear, inertial dynamics of the robot; uncertainties in hydrodynamic parameters; and unmodeled ocean current disturbances [3]. ROVs have this issue intrinsically solved by offloading most control to the human operator. It is because of this that UUV control methods are, for the most part, only relative to AUV and TUV systems. To a limited extent, ROVs require some automatic control for station keeping or depth tracking, but nothing as complex as that required for AUVs and TUVs. The following subsections attempt to outline the main types of control paradigms in use today, and when they are applicable.

#### **Close Control**

Since the ROV platform provides instantaneous control and feedback, the only navigational limitation is the human's ability to pilot the vehicle. This type of control method is called close control. With close control, a human operator has direct control of almost every system on the remote vehicle. This makes it ideal for tasks requiring precise interaction with the vehicle such as work site inspection, wreck investigation, filmmaking, mine countermeasures and anything requiring manipulation of remote objects. Preferably, close control would be applied to a TUV over its complete operating range. Unfortunately, the varying time delay inherent with the acoustic link prevents close control beyond a certain range (discussed in §2.1). Similarly, close control cannot be used with AUVs since they do not facilitate the constant communications needed.

### Supervisory Control

When close control is not practical, operators must defer some, or all, of the precise control to other methods. Supervisory control is useful for long-range exploratory missions or those that do not require meticulous control. Ballou [16] found that beyond a certain range, close control was impossible due to the control delay of the acoustic link. At the maximum time delay for his application (1.5 s), Ballou gave the vehicle a limited degree of autonomy allowing it to perform basic maneuvering tasks on its own. In addition to autonomy, Ballou also restricted the response of the remote vehicle to compensate for the delay time. This is similar to the method proposed by Slawiñski *et al.* [26]. Here, they introduce a "fictitious force", that both limits the vehicle's response while automatically avoiding obstacles. The "fictitious force" augments the commands sent by the operator during periods of large time delays or potential collisions.

Autonomous control is even more prevalent in AUV systems where there is no provision for close control. AUVs must rely solely on their own autonomy and preprogrammed missions, carried out with on-board autopilot algorithms. Usually, this type of mission is created by a human operator using a top-down type of mission planning software (as shown in Figure 1.10). The operator designs the mission by defining waypoints, and can monitor the progress of the vehicle while the autopilot carries out the mission. By far, the most popular type of autopilot is a PID controller [21, 23, 26–31] and variations thereof. PID controllers are popular due to their ease of implementation, robustness, and acceptable performance when compared to most other automatic controllers. The operator needs only to send the desired state, or waypoint, to the controller and it will regulate each thruster on the vehicle until that state is achieved. The use of a PID controller for an underwater autopilot system is discussed at length in §2.4.2.

The main disadvantage of PID controllers is poor performance when faced with large time delays or very high-order systems. The time delay problem can be remedied by placing the controller on the remote vehicle itself [32]. This exploits the high-bandwidth connection afforded by on-board systems for the autopilot processing. The poor performance caused by high-order systems cannot be so easily fixed, so many improvements and alternatives to PID controllers exist. One such improvement, proposed by Guo *et al.* [33], introduces a nonlinear compensating term. This term is manifested as an additional force added to the normal PID commanded force and is used to compensate for the nonlinearities of the

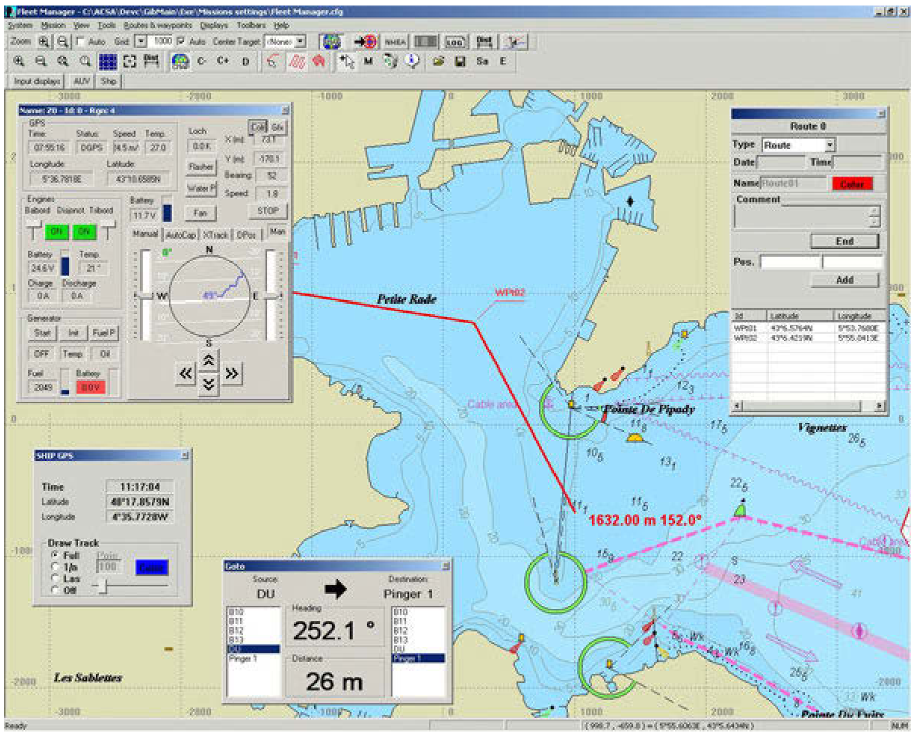


**Figure 1.10:** A top-down type of mission planning software called *FleetManager* (*source*: `http://www.underwater-gps.com/uk/product-detail.php?pr_id=9`).

dynamics and unmodeled disturbances.

Departing from the PID controller altogether, one can implement a model predictive controller (MPC) [34, 35]. MPCs try to predict the future state of the system using the current state and inputs. Then, a parameter optimization technique is used until the predicted state matches a target state. For example, Naeem *et al.* [35] implement a genetic algorithm that minimizes a cost function to determine the control sequence necessary to follow a predetermined trajectory.

Another popular controller is a sliding mode controller (SMC), which is useful for nonlinear systems since they are composed of varying control structures [36]. Based on the vehicle's current state, the controller slides along a control surface, switching between the different structures. This switching action can create a problem called control chattering where the controller quickly switches between different modes, causing large discontinuities in the output. This can damage actuators or cause unexpected dynamics. Chatchanayuenyong and Parnichkun [37] eliminate chattering by introducing a PI controller during the switching phases. With this method, the good steady-state response of PI controllers is combined with the superior transient response and robustness to uncertainties of the SMC.

Finally, Aguiar *et al.* [9] introduce a custom adaptive controller derived from the kinematic model of the vehicle. This controller is derived from first principles and extended to the dynamic case using integrator back-stepping and Lyapunov-based techniques. The controller is shown to perform very well in the face of unknown ocean currents and model parameter uncertainties.

## 1.2 The Problem With Current UUV Systems

There is a need for a new class of underwater vehicle that can perform both close control tasks as well as long-range exploration without manual reconfiguration. The two popular classes of underwater vehicles cannot fulfill this need easily: ROVs are limited in range and cost by their tether and AUVs cannot perform complex tasks reliably. Hovering AUVs attempt to combine some aspects of ROVs with AUV systems, but are limited to short ranges and specialized tasks. The third, less popular UUV class, TUVs, show potential but have not yet been exploited to their full capabilities. The ability to maintain constant communication through an acoustic link introduces the drawback of a time-varying, control time delay. Consequently, the close control method central to ROV systems cannot be directly applied to TUVs. The varying time delay makes direct human control impossible beyond a certain range.

Ballou's [16] solution to this was separated into three fundamental ideas. First, he would try to minimize the distance between the TUV and the transmitter at all times, thus reducing the delay. Second, in cases of higher delays, he decreased the responsiveness of the TUV to match the delay time. This is similar to the solution proposed by Bulich *et al.* [39], where they force the vehicle into a "move-and-wait" control strategy (discussed in §2.1.2) when the delay induced by the internet connection is large. Lastly, the vehicle was also required to perform simple tasks autonomously to compensate for large delays. While this method works, it lacks the ability to smoothly transition between the different control paradigms. It is also preferable for the controller to maintain the full responsiveness of the vehicle for all time delays, instead of throttling it. None of the control schemes of AUVs can directly solve the problem either. AUV control systems mainly focus on long-range and exploration missions with no capacity for close control. The hovering AUVs proposed by Beaujean and Carlson [11] and Vaganay *et al.* [15] attempt to introduce operator control in combination with autonomous control but are limited to very short ranges. As such, they did not need to deal with the time delay issue.

To summarize, the problem is twofold: first, replacement of the tether with an acoustic link introduces the impediment of a variable time delay; second, no existing control system can easily and smoothly transition between the two types of operating modes. A control system is needed that can combine the close control capabilities of ROVs with an AUV's proficiency in exploration (as illustrated in Figure 1.11). This research proposes a new continuum of control that evolves through existing control paradigms, gradually introducing more autonomy as range, and therefore time delay, increases. As explained above, the use of a TUV system can reduce operating costs anywhere from 50–90%. This is a huge cost savings and if the controller can accommodate all required tasks, the TUV has significant potential. The work of Bulich *et al.* [39] also suggests that this type of controller may also be useful for tethered ROV systems that are controlled through an internet link that induces its own delays. With current and future acoustic communication systems allowing real-time video and control for many kilometers, the author feels that the communications link, although the main bottleneck, will support this type of application

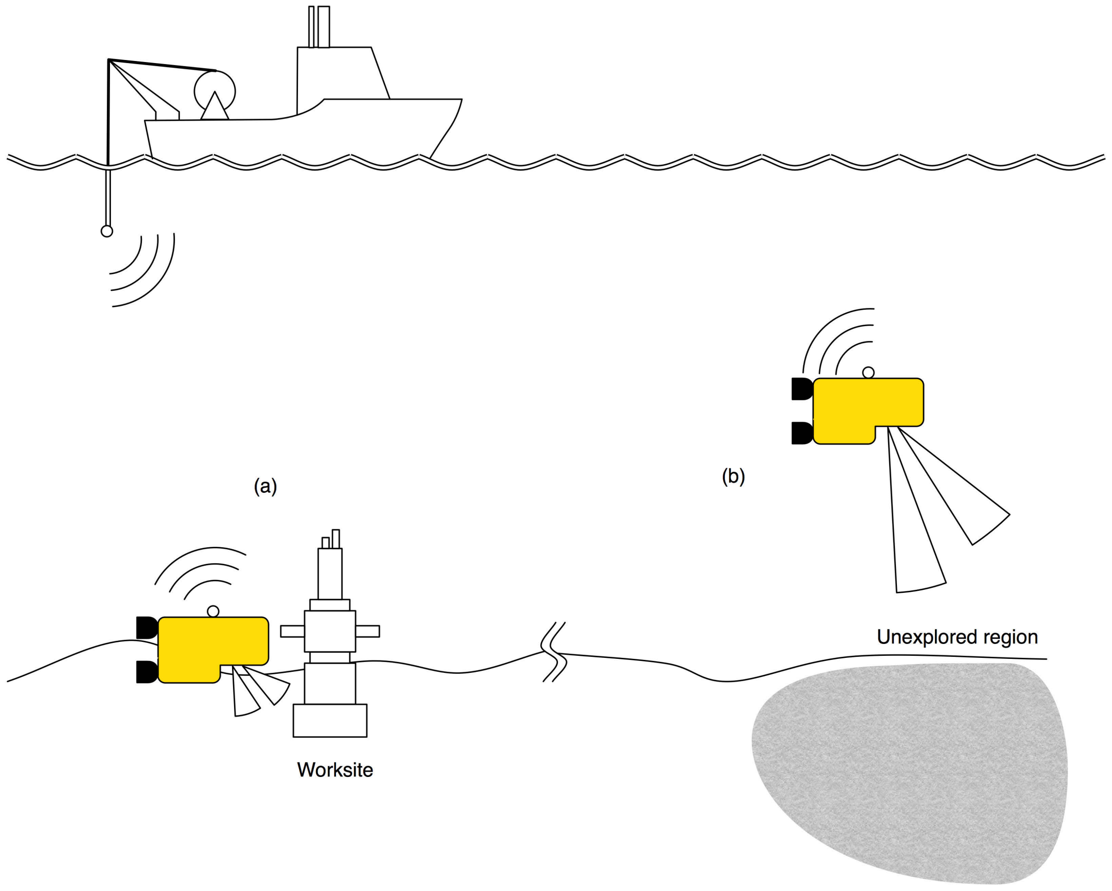


**Figure 1.11:** Envisioned use for complete TUV system: (a) the TUV is used for close worksite inspection tasks; (b) the TUV seamlessly transitions to exploration mode to travel to, or find, additional worksites.

on the TUV vehicle platform.

### *1.2.1 Research Objectives*

With respect to the problem statement, this research attempts to fulfill a number of objectives:

1. To design a controller that can handle both close and supervisory control paradigms and smoothly transition between them, forming a complete control continuum.
2. To analyze the feasibility of the controller given current and future technology.
3. To build a simulation capable of accepting commands from a human operator and providing real-time, visual feedback from a simulated TUV vehicle in an underwater environment.
4. To implement and test the control continuum in the simulator.
5. To analyze the performance of the continuum by designing a representative task for the operator to carry out in the simulated environment.
6. To suggest future work for the control continuum in order to carry it forward and implement it on an actual vehicle.

The research will be considered a success if the above objectives are met and the implemented control continuum allows the operator to efficiently control the vehicle over a wide range of operating distances and time delays.

## 1.3 Thesis Overview

Chapter 2 develops the theory necessary to create a controller that can accomplish the research objectives. The problem with human control through time delay is outlined and a solution proposed. Specifically, a control continuum for the vehicle is designed that isolates the operator from the delay using predictive and autopilot controllers. Chapter 3 presents a simulation framework to test the proposed control continuum. Using the simulation environment, each component of the control continuum is tuned for the given test scenario. A test for a human operator is also devised consisting a simulated obstacle course. Finally, Chapter 4 analyzes the performance of the control continuum in the simulated environment with human operators. The results of the tests are analyzed and conclusions are drawn about the effectiveness of the control continuum.

## 1.4 Contributions

In order to meet the research objectives, this thesis offers the following contributions:

1. A control continuum was designed and implemented for a TUV system. The main component of the continuum, the auto-op command mixer, enables the smooth evolution through different control paradigms based on the control time delay. This enables the operator to remain in close control with the TUV vehicle for an extended range.

2. An *ad hoc* tuning method is presented for the control continuum. It allows the control to remain stable for a wide range of operating distances and allows the operator to adapt the tuning to the given task.

3. A TUV simulator environment was designed and implemented employing two simulated vehicles in an underwater scene. The simulator also introduces a simulated control time delay to test the efficiency of the control continuum. The simulator is easily adapted to match different vehicle configurations and allows the continuum to be tuned on demand. It also presents a testing framework with a virtual obstacle course for human navigation and the ability to measure their performance.

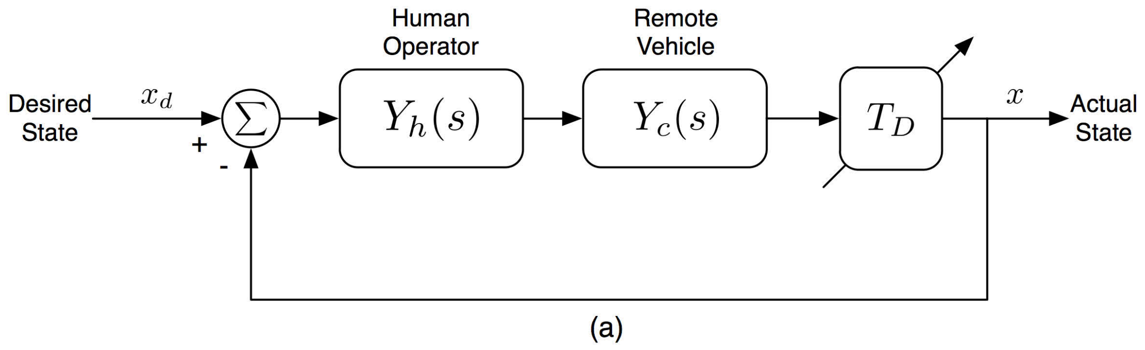
# CHAPTER 2

# THEORY

To develop an effective controller, we must first analyze and understand the system we wish to control. This section presents the theoretical foundation necessary for designing a practical controller for the TUV vehicle platform. First, the problem with the time delay is stated and justified. Next, the coordinate system is presented and the vehicle model is discussed. With respect to the vehicle model, both a predictive display and a PID autopilot are designed to help cope with the communications time delay. Finally, the control continuum is formulated and the mechanism for the evolution through it is described and analyzed.

## 2.1 Human Control Through Time Delay

As presented in §1.1.2, the main problem introduced with the TUV acoustic link is the time-varying control time delay. This time delay can be modeled as one lump delay element in the forward path of the control loop, as shown in Figure 2.1(a). In actuality, this delay would be split in half between the forward and reverse paths; however, from the point of view of the controller, all delay elements may be lumped together [38]. This is in contrast to a typical ROV loop, as shown in Figure 2.1(b). Usually, ROVs have no delay element in the control loop, supporting real-time, manual control throughout the entire range of the ROV. In special cases, such as the internet-controlled ROV discussed by Bulich *et al.* [39], a time delay element may exist in the order of several seconds. The following sections discuss the problems imposed by the delay element with respect to the



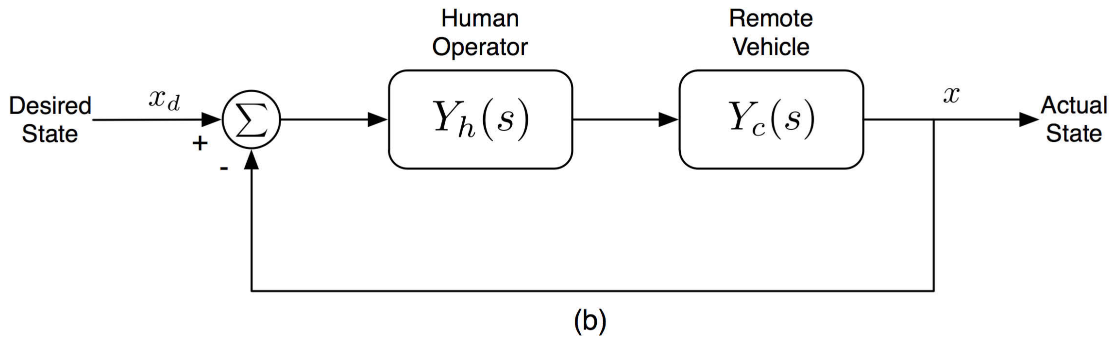


**Figure 2.1:** Illustration of two control loops: (a) typical control loop of a TUV system with a time-varying time delay element, $T_D$, in the feedback path; (b) the control loop of an ROV with no delay element.

human operator. A method to circumvent the delay is also presented.

### *2.1.1 The Problem With Time Delay*

When operating remote devices, such as underwater vehicles, humans attempt to construct an internal dynamic model of themselves and the environment [40]. The internal model can be used to both plan and predict the state of the remote vehicle and its environment. This provides the human's unique ability to adapt to rapidly changing environments while still maintaining control of the vehicle and guiding it to the goal by making impromptu corrections to their internal model. A problem arises when faced with nonlinear and relatively high-order systems, like underwater vehicles. With systems such as these, the human's internal model can no longer accurately predict the future state given current stimuli, and the ability to control the vehicle breaks down. The problem is exacerbated when adding a time delay into the loop since the human is tasked with extending their internal prediction even further ahead in time. The difficulty of prediction is also increased since, with pure delay, the human cannot use the current derivatives (vehicle speed) to help with the prediction, as is the case with systems with only a slow response time. This problem is illustrated in Figure 2.2: (a) shows the case where the current system derivatives can be used to predict the future of the slow system response one time constant ($\tau$) ahead; whereas (b) shows the case where, since the feedback is delayed, the derivatives are not available until the time delay ($\tau_d$) has elapsed.

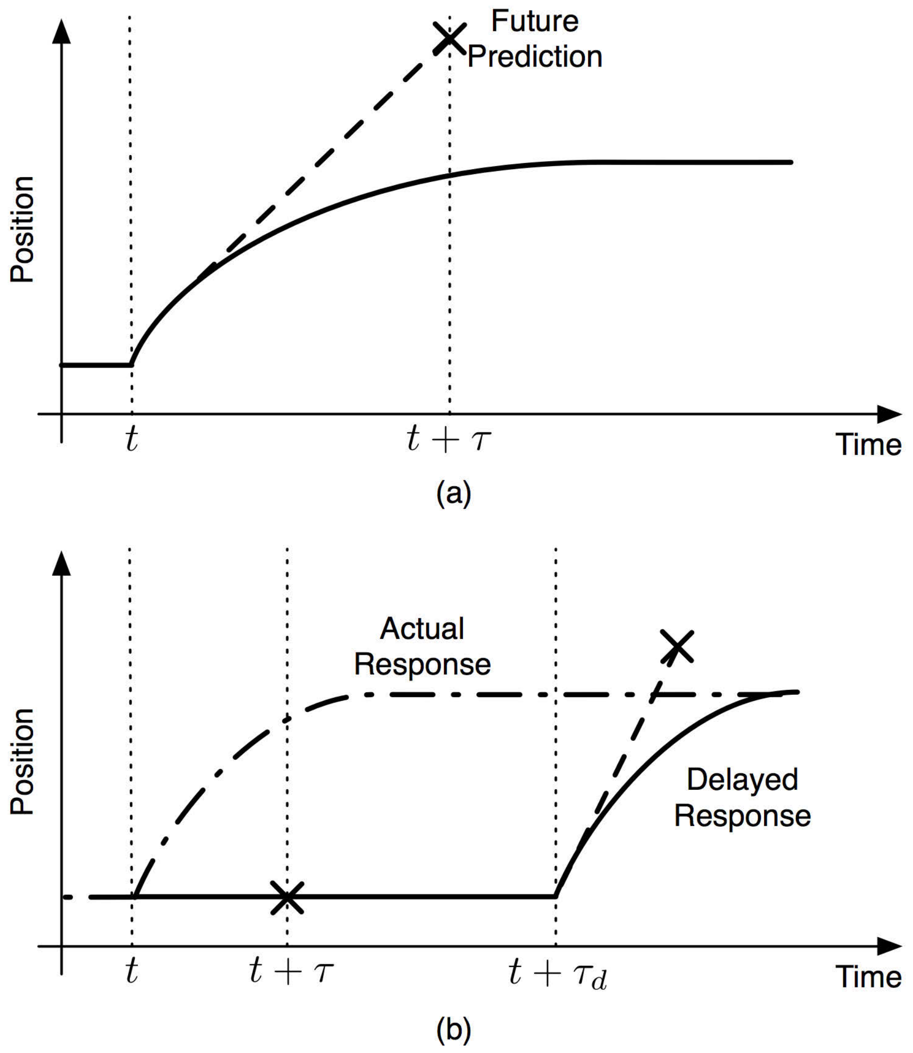


**Figure 2.2:** The effect of time delay on human prediction accuracy: (a) a system with no transport lag, but large system response time, $\tau$; (b) A system with relatively small response time but a large transport lag, $\tau_d$.

Cheng [40] analyzed this problem by building a simple model of the human operator and the environment. With respect to Figure 2.1, a simplified model of the human operator, for inertial systems, can be written in the Laplace domain as:

$$Y_h(s) = K_h(T_L s + 1)e^{-s\tau_e} \tag{2.1}$$

where $\tau_e$ is the effective time delay of the human operator, including the pure time delays associated with reading the displays and deciding on the required control action. $K_h$ is the gain that can be adjusted by the operator, along with $T_L$, the lead time constant. These parameters are constantly and internally adjusted to better match the controlled system to achieve stable closed-loop control. For illustration purposes, Cheng anaylzes a simple inertial system with dynamics:

$$F = m\ddot{x} + b\dot{x} \tag{2.2}$$

where $F$ is the thrust, and $m$ and $b$ are the vehicle mass and damping coefficient, respectively. In the Laplace domain, the impulse response of (2.2) can be written as:

$$Y_c(s) = \frac{K_c}{s(\tau s + 1)} \tag{2.3}$$

where $K_c = \frac{1}{b}$ and $\tau = \frac{m}{b}$ are the gain and time constant of the controlled system.

The lead compensator term in the human model (2.1), $T_L s + 1$, represents the weight the operator places on the vehicle's velocity, as opposed to its position [38]. This lead term is actually used by the human operator to internally predict the future response of the controlled vehicle. This is useful for inertial systems (like underwater vehicles) that produce a relatively large, lag time constant. With this type of system, the operator adjusts their internal lead time constant, $T_L$, to match the time constant of the controlled system according to the open-loop cross-over model [38]:

$$Y_h Y_c \approx \frac{w_c e^{-s\tau_e}}{s} \tag{2.4}$$

where $w_c = K_h K_c$ is the cross-over frequency. In order for this condition to be satisfied, the human must adjust their lead time constant to match the system time constant ($T_L = \tau$). The compensation effectively allows the operator to control the predicted state of the vehicle, instead of the lagging, current state. From Figure 2.1(b), and assuming the operator is trying to pilot the vehicle to a desired point, the lead compensator control action is proportional to the future error and can be written as $(\tau s + 1)(x_d - x)$ and is illustrated in Figure 2.3. Here, the operator tries to pilot the vehicle along the desired path, $x_d$, through controlling the actual vehicle's path, $x$, by attempting to minimize the error one time constant ahead using the current speed for the prediction.

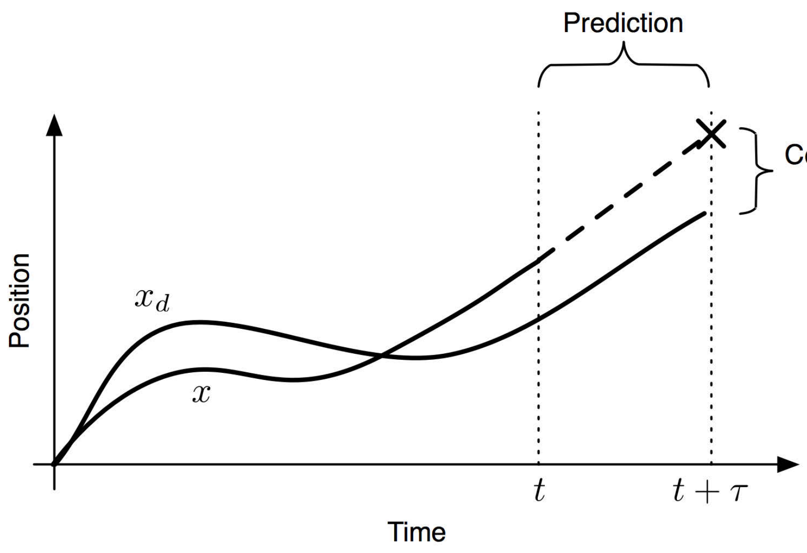


**Figure 2.3:** Illustration of internal human prediction, using the lead compensator term, to pilot the vehicle to a desired location (based on a figure in [40]).

As the process' time constant increases, the human operator must correspondingly increase their lead time constant. This greatly escalates the mental loading felt by the operator since they need to predict the response further into the future [38–40]. When a communications delay is added, such as $T_D$ in Figure 2.1(a), then the human is forced to predict into the future even further. The open-loop transfer function, with added delay, then becomes:

$$H(s) = K_h(\tau s + 1)e^{-s\tau_t}Y_c(s) \tag{2.5}$$

with,

$$\tau_t \triangleq \tau_e + \tau_d$$

where $\tau_d$ is the time constant of the pure delay element, $T_D$, introduced by the acoustic link in Figure 2.1(a). As shown, the added delay can be lumped in with the delay associated with the operator's transfer function (2.1) [38]. When this delay reaches ⅓ s, the control tends to destabilize [41]. When the total delay reaches 1 s or greater, then direct control is almost impossible. Some research [39] suggests humans can handle delays up to 1.5 s; however, with delays so high, the operator is forced to use the "move-and-wait" strategy and with the resulting significant mental loading, the slightest distraction can break their concentration. Using the vehicle model (2.3), the maximum human operator gain can be calculated from (2.5) as:

$$K_h^{\max} = \frac{\pi}{2\tau_t K_c} \tag{2.6}$$

which represents how hard the operator can drive the system before it becomes unstable.

Using the 1st-order Padé approximation [42, 43], the pure time delay transfer function can be written as:

$$e^{-s\tau_t} \approx \frac{-s + {}^{2}\!/\!_{\tau_t}}{s + {}^{2}\!/\!_{\tau_t}} \tag{2.7}$$

The effect of varying the delay is illustrated with the closed-loop root-locus of (2.5) in Figure 2.4. Here, as the delay increases, the root-locus is pushed toward the right half-plane causing the control system to become unstable. As a result, as $\tau_t$ increases, the operator must become increasingly conservative with their corrections in order to maintain control of the vehicle. With the time delay of a TUV system varying from almost zero to several seconds over the range of operations, direct human control is not feasible without some assistance.

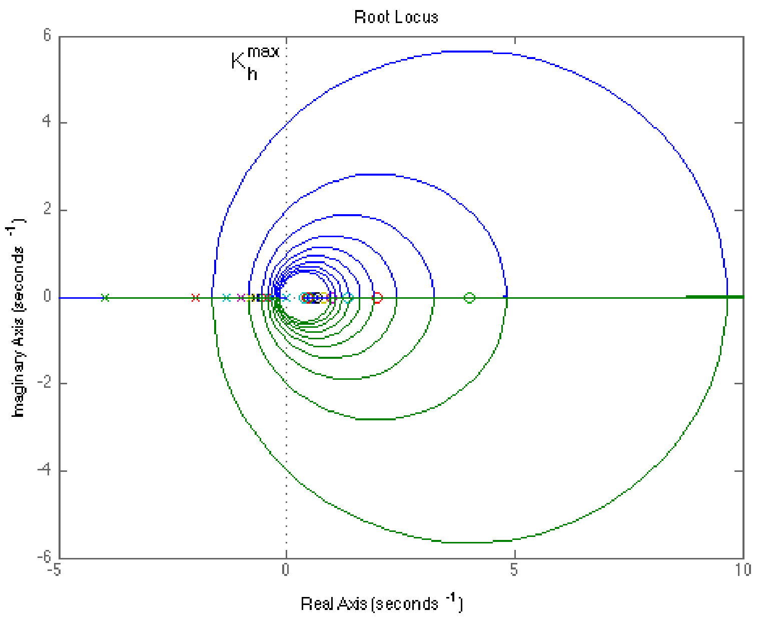


**Figure 2.4:** Family of root-loci of the system (2.5) with a varying total time delay.

### 2.1.2 *Operator Isolation*

Since the time delay is an intrinsic property of the acoustic channel, improving the link will do nothing to abate the delay. Control systems must therefore exploit other methods to allow for direct operator control. One such method is the so-called "move-and-wait" control strategy [32, 41]. That is, the operator issues only short commands and waits the entire delay time, $T_D$, to receive the response before issuing more commands. This corresponds to decreasing the human gain, $K_h$, so as to avoid moving into the unstable region in Figure 2.4. The drawback is greatly reduced operator efficiency making this strategy impractical for complex missions.

#### Predictor Displays

A more effective strategy is to isolate the operator from the delay completely. This can be accomplished through the introduction of a predictor display [40, 44] as illustrated in Figure 2.5. These systems have been shown to greatly assist an operator in controlling high-order, nonlinear systems that have little damping. The reason for this is the human model, as presented in §2.1.1, cannot adapt itself adequately to match the system model. Predictor displays aid the operator by constructing an external prediction of the system and allow the operator to control this output instead of the vehicle's output directly. This is especially advantageous for systems with a pure delay element since the predictor can, effectively, remove the delay allowing the operator to control an undelayed vehicle model and maintain the illusion of close control.

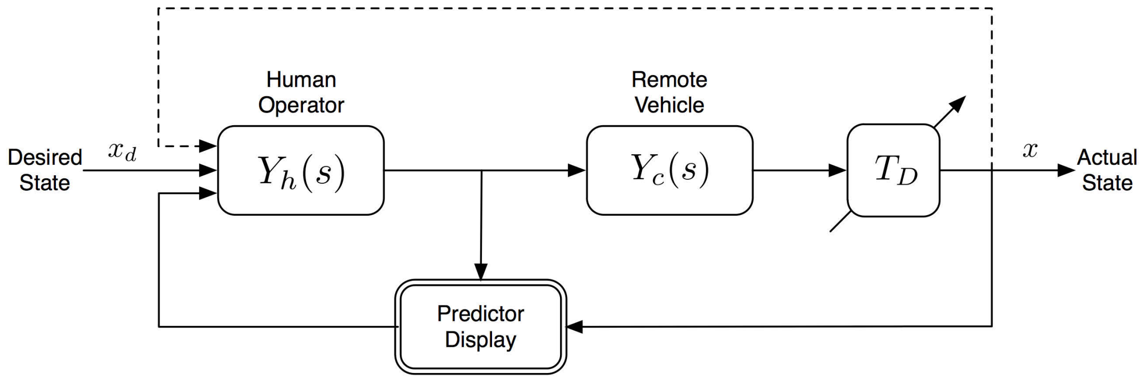


**Figure 2.5:** Block diagram of a predictor display used for operator isolation. The upper dashed line indicates that the operator may still have access to the actual, delayed feedback from the vehicle.

There are two types of predictor displays. The simpler version utilizes the Taylor series expansion to perform an extrapolation based on the current state and time derivatives.

Assuming the vehicle response is represented by $y(t)$ and the prediction span is $h$, the Taylor series extrapolation can be written as [40]:

$$y(t_0 + h) = y(t_0) + hy'(t_0) + \frac{h^2}{2}y''(t_0) + \cdots \tag{2.8}$$

This method is used to predict the state of the vehicle at $t_0 + h$ given only the current system output at $t_0$. As a result, no knowledge of the controlled system is needed, making the Talyor series extrapolation easy to implement and applicable to any system. The major disadvantage is the low prediction accuracy. Since the linear extrapolation is solely based on current state and derivatives, any changes in input or disturbances will not be accounted for in the prediction. Additionally, the Taylor series expansion only holds for prediction spans that are orders of magnitude below the system time constant, that is $h \ll \tau$. The limited range makes the Taylor series extrapolation alone inadequate for the range of delays associated with a TUV system.

The other type of predictor display uses the fast-time model method [40]. Here, a mathematical model of the system is run many times faster than real-time to predict the future state. As a result, this method has several advantages over the extrapolation method. First, changing inputs can be accounted for during the prediction span and incorporated into the model. Next, any system nonlinearities (such as saturation effects) can be accounted for by the model and factored into the prediction. Most importantly, this method affords a greater prediction span that is only limited by the fidelity of the model employed. The major limitation is this dependence on the model, however. If there are unmodeled disturbances, then the fast-time prediction will produce less accurate results as the prediction span increases. Secondly, this method can be computationally expensive unless care is taken to employ simplified models (such as the one discussed in §2.3.6).

To show how the fast-time method is able to assist the operator, it is helpful to once again examine the Laplace transform. The predictor works first by reducing the effect of the pure delay element, $T_D$, in Figure 2.5. If the prediction span is denoted $h$, the predictor will introduce a compensating term, $e^{sh}$, and the transfer function of the delay element can then be written as:

$$T_D{}'(s) = e^{s(h-\tau_d)} \tag{2.9}$$

When the prediction span is equal to the time delay, $h = \tau_d$, the delay associated with the acoustic link element is dropped from the system completely. In this case, $\tau_t$ in (2.5) is reduced to the human response time only, $\tau_t = \tau_e$, thus increasing the maximum control gain (2.6) and increasing the control stability.

Predictor displays can also be used to compensate for slow system response times. If the time response of the vehicle is denoted $y(t)$, the the fast-time model is represented by a time scaling, $y(\alpha t)$, with $\alpha > 1$. The Laplace transform of this time scaling is written as:

$$\mathcal{L}\{y(\alpha t)\} = Y'(s) = \frac{1}{\alpha} Y\left(\frac{s}{\alpha}\right) \tag{2.10}$$

When applied to the system in Figure 2.5 the vehicle model transfer function (2.3) becomes:

$$Y_c'(s) = Y_c\left(\frac{s}{\alpha}\right) = \frac{K_c \alpha}{s\left(\frac{\tau}{\alpha}s + 1\right)} \tag{2.11}$$

Correspondingly, the human operator must also adjust their internal model to match the accelerated one such that [40]:

$$Y_h{}'(s) = K_h{}'\left(\tau' s + 1\right) e^{-s\tau_e{}'} \tag{2.12}$$

where $\tau' = \frac{\tau}{\alpha}$ is the reduced time constant from (2.11), $\tau_e{}'$ is the reduced reaction time, and the operator gain, $K_h{}'$, is also reduced by a factor of $\alpha$.

Predictor displays remove the effect of the pure delay element, reducing the extent of the internal prediction a human must perform. This reduction greatly relieves the mental loading felt by the operator [38, 40]. The advantages of these displays are then threefold:

- The mental loading of the operator is significantly reduced.
- The learning period is shortened since operators do not have to learn how to cope with the delay manually.
- The operator can achieve better control performance since they do not need to adopt the "move-and-wait" control strategy.

**Vehicle Autonomy**

Another method of operator isolation is to remove the need for the operator completely by introducing vehicle autonomy. Various forms of autonomous controller exist as outlined in §1.1.4. These controllers do not suffer from the effects of transport lag since they are usually completely contained on the vehicle itself. However, some autonomous controllers such as model predictive controllers, can still use prediction to compensate for large system time constants.

## 2.2 Kinematics and Coordinate System

Many different standard notations and representations exist for underwater vehicles. For simplicity, the coordinate systems chosen for this thesis are a combination of those proposed in [21, 27] with some modifications to better reflect the chosen simulation environment (described in §3.3).

Two frames of reference are required to fully represent the vehicle motion in a 6-DOF (degree of freedom) environment. First, a body-fixed reference frame is attached to the moving vehicle with its origin at the centre of gravity of the vehicle. The axes are fixed with respect to the vehicle's principal axes of inertia and are written as $x$–$y$–$z$. To better match the simulation environment, the $x$-axis is aligned with the forward (or surge) direction of the vehicle, the $y$-axis with the vertical (or heave) direction, and the $z$-axis with the lateral (or sway) direction. This reference frame is useful for representing the body-relative velocities of the vehicle. The second reference frame is an Earth-fixed, inertial frame. This frame is required to represent the position of the vehicle and is written as $X$–$Y$–$Z$ with its origin at some known location. The orientation of an underwater vehicle is expressed relative to the Earth-fixed frame through a composition of three intrinsic rotations expressed as the Euler angle set $Y(\psi)$–$Z'(\theta)$–$X''(\phi)$, where $\psi$, $\theta$, and $\phi$ are the yaw, pitch, and roll angles, respectively. Both the position and orientation of the vehicle are written in a single vector in the inertial frame [21]:

$$\boldsymbol{\eta} = \left[ \begin{array}{cc} \overbrace{x \quad y \quad z}^{\boldsymbol{r}} & \overbrace{\phi \quad \psi \quad \theta}^{\boldsymbol{\beta}} \end{array} \right]^T \tag{2.13}$$

where $\boldsymbol{r}$ and $\boldsymbol{\beta}$ are the distance and orientation vectors, respectively. Similarly, the velocity vector, in the vehicle-fixed frame, is written as:

$$\dot{\boldsymbol{\zeta}} = \left[ \begin{array}{cc} \overbrace{u \quad v \quad w}^{\boldsymbol{v}} & \overbrace{p \quad q \quad r}^{\boldsymbol{\omega}} \end{array} \right]^T \tag{2.14}$$

with $\boldsymbol{v}$ and $\boldsymbol{\omega}$ as the linear and rotational velocities, respectively. This representation of the

TUV is illustrated in Figure 2.6. The transformation of velocities from body-fixed frame to the inertial frame is written as:

$$\dot{\boldsymbol{\eta}} = \mathbf{T}_\beta \dot{\boldsymbol{\zeta}} \tag{2.15}$$

where $\mathbf{T}_\beta$ is a rotation matrix based on the current orientation, $\beta$, similar to those derived in [21, 27], and is composed of linear, $\mathbf{T}_1$, and angular, $\mathbf{T}_2$, velocity transformations:

$$\mathbf{T}_\beta = \left[\begin{array}{c|c} \mathbf{T}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{T}_2 \end{array}\right] \tag{2.16}$$

## 2.3 Vehicle Model

With a suitable representation of the vehicle's coordinate system, a model can be constructed for the dynamics of the vehicle. It is instructive to first present the overall model, then discuss each constituent part and finally, decide on a simplified model. Following the approaches in [21, 27, 45]. the complete vehicle model is written as:

$$(\mathrm{M} + \mathrm{M}_A)\ddot{\boldsymbol{\zeta}} = \boldsymbol{F}_B(\boldsymbol{\eta}) + \boldsymbol{F}_D(\dot{\boldsymbol{\zeta}}_{\text{rel}}) + \boldsymbol{F}_C(\dot{\boldsymbol{\zeta}}) + \boldsymbol{u}(t) \tag{2.17}$$

where $\mathrm{M}$ and $\mathrm{M}_A$ are the mass and added mass matrices; $\boldsymbol{F}_B, \boldsymbol{F}_D$ and $\boldsymbol{F}_C$ are the buoyancy, drag, and Coriolis forces; and $\boldsymbol{u}(t) = \boldsymbol{F}_T(t) + \boldsymbol{\epsilon}(t)$ is the thruster, and unmodeled disturbance (such as ocean current) external forces. As indicated by Steinke [21], the drag force, $\boldsymbol{F}_D$, is a function of the vehicle's velocity relative to the water, $\dot{\boldsymbol{\zeta}}_{\text{rel}}$. Assuming an ocean current velocity vector, $\boldsymbol{v_c} = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T$, in the inertial frame, the relative velocity, in the body-fixed frame is:

$$\boldsymbol{v}_{\text{rel}} = \boldsymbol{v} - \mathbf{T}_1{}^T \boldsymbol{v_c} = \begin{bmatrix} u_{\text{rel}} & v_{\text{rel}} & w_{\text{rel}} \end{bmatrix}^T \tag{2.18}$$

Then, the relative velocity vector from (2.17) is written as:

$$\dot{\boldsymbol{\zeta}}_{\text{rel}} = \begin{bmatrix} \boldsymbol{v}_{\text{rel}} \\ \boldsymbol{\omega} \end{bmatrix} \tag{2.19}$$

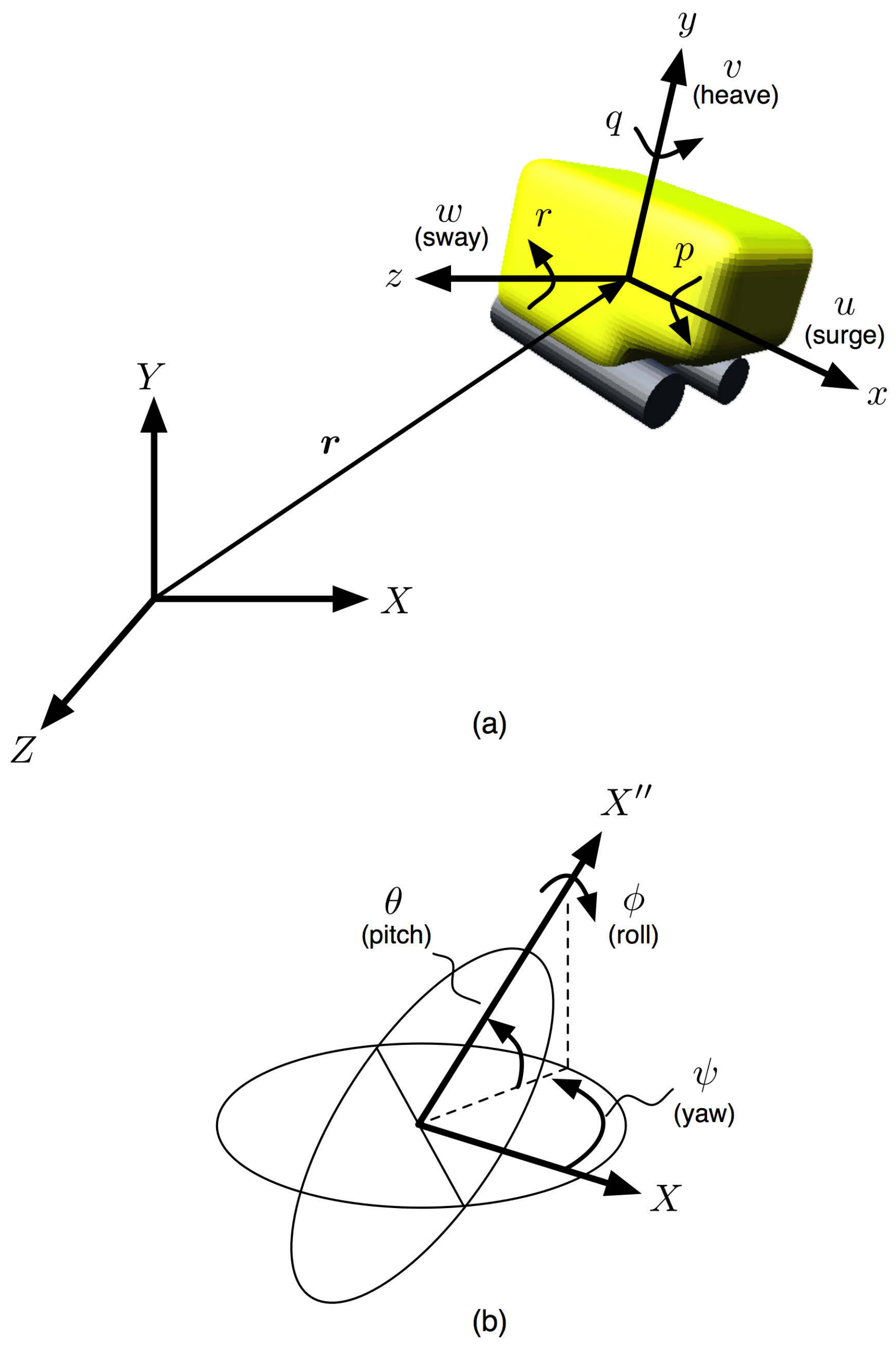


**Figure 2.6:** (a) The two coordinate systems used to describe the motion of the TUV. The $X$–$Y$–$Z$ inertial frame is used for position and orientation, while the body-fixed frame, $x$–$y$–$z$, is used for velocity. (b) The representation of orientation as a composition of three intrinsic rotations relative to $X$ in the inertial frame and ending in $X''$.

### *2.3.1 Mass, Added Mass and Inertia*

The mass matrix, $\mathbf{M}$, contains both the mass and inertial parameters of the vehicle in all DOFs:

$$\mathbf{M} = \left[\begin{array}{c|c} \mathbf{m} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right] \tag{2.20}$$

with

$$\mathbf{m} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix}$$

$$\mathbf{I} = \begin{bmatrix} I_x & I_{xy} & I_{xz} \\ I_{xy} & I_y & I_{yz} \\ I_{xz} & I_{yz} & I_z \end{bmatrix}$$

Matrix $\mathbf{m}$ represents the mass of the vehicle, which is equal for all axes. Matrix $\mathbf{I}$ represents the inertia tensor of the vehicle. A simplification can be made to this if the body-fixed axes are chosen to coincide with the vehicle's axes of symmetry [27]. This diagonalizes the inertia tensor and can then be written as:

$$\mathbf{I} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{2.21}$$

The moment inertia along a given axis is calculated by the volume integral:

$$I = \int_V \rho(\boldsymbol{r})\, \mathrm{d}(\boldsymbol{r})^2\, dV(\boldsymbol{r}) \tag{2.22}$$

where $\rho(\boldsymbol{r})$ is the density at point $\boldsymbol{r}$, and $\mathrm{d}(\boldsymbol{r})$ is the distance from the axis to point $\boldsymbol{r}$. For this research, the inertial body will be assumed to be a box of dimensions $a, b,$ and $c$, as shown in Figure 2.7. Then, the moment of inertia for each body-fixed axis, assuming constant density, is:

$$I_x = \rho \int_V \left(y^2 + z^2\right) dx\, dy\, dz = \frac{m}{12}\left(b^2 + c^2\right)$$
$$I_y = \rho \int_V \left(x^2 + z^2\right) dx\, dy\, dz = \frac{m}{12}\left(a^2 + c^2\right)$$
$$I_z = \rho \int_V \left(x^2 + y^2\right) dx\, dy\, dz = \frac{m}{12}\left(a^2 + b^2\right)$$

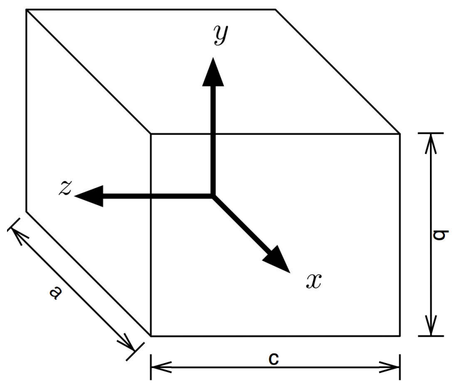


**Figure 2.7:** Box used to calculate the inertia tensor of the TUV along the $x$-, $y$-, and $z$-axes.

The added mass term, $\mathbf{M}_A$, is often misunderstood as a finite amount of water that gets "stuck" to the vehicle, adding additional mass [27]. It should be more precisely understood as a pressure-induced force resulting from the motion of the body through the water, proportional to the acceleration of the body. A simple, decoupled model of this can be written as [21]:

$$\mathbf{M}_A = \text{diag}(x_{\dot{u}}, y_{\dot{v}}, z_{\dot{w}}, l_{\dot{p}}, m_{\dot{q}}, n_{\dot{r}})$$

For more information on added mass refer to [21, 27] and the references therein. For this thesis, since no actual vehicle is available and added mass depends on vehicle configuration and structure, the term will be omitted from the model, as in [31].

## *2.3.2 Buoyancy*

Usually, underwater vehicles such as ROVs and TUVs are equipped with a foam pack. This foam pack creates a buoyancy force that, combined with gravitational forces, produces restoring forces and moments that serve to self-right the vehicle in the pitch and roll DOF. It will be assumed the centre of buoyancy is offset only in the $y$-axis, or vertical direction, of the body-fixed frame from the centre of mass. Therefore, in the body-fixed frame, the buoyancy force is written as [27]:

$$\boldsymbol{f}_B = \mathbf{T}_1{}^T \begin{bmatrix} 0 & f_B & 0 \end{bmatrix}^T \tag{2.23}$$

located, in the body-fixed frame, at:

$$\boldsymbol{r}_B = \begin{bmatrix} 0 & y_B & 0 \end{bmatrix}^T \tag{2.24}$$

This thesis assumes the vehicle will be trimmed to be neutrally-buoyant, so the buoyancy force is equal to the gravitational force. Therefore, the buoyancy force creates only self-righting moments and the buoyancy load of (2.17) is calculated, in the body-fixed frame, as:

$$\boldsymbol{F}_B(\boldsymbol{\eta}) = \begin{bmatrix} \mathbf{0} & \boldsymbol{r}_B \times \boldsymbol{f}_B \end{bmatrix}^T \tag{2.25}$$

## 2.3.3 Drag

An exact model of hydrodynamic drag for an underwater vehicle would be highly nonlinear and coupled in all 6 DOF. Instead, a simplified model as in [21, 27, 45] decouples each DOF and ignores any terms higher than second-order:

$$\boldsymbol{F}_D(\dot{\boldsymbol{\zeta}}_{\text{rel}}) = -\mathbf{D}_L\dot{\boldsymbol{\zeta}}_{\text{rel}} - \mathbf{D}_Q\dot{\boldsymbol{\zeta}}_{\text{rel}}\left|\dot{\boldsymbol{\zeta}}_{\text{rel}}\right| \tag{2.26}$$

where $\mathbf{D}_L$ and $\mathbf{D}_Q$ are the linear and quadratic drag matrices. Since a decoupled model is assumed, the drag matrices are diagonal and the drag for each DOF has the form:

$$F_D = -k_\zeta\dot{\zeta}_{\text{rel}} - k_{\zeta|\zeta|}\dot{\zeta}_{\text{rel}}\left|\dot{\zeta}_{\text{rel}}\right| \tag{2.27}$$

where the drag coefficients, $k_\zeta$ and $k_{\zeta|\zeta|}$, need to be identified for each DOF.

## 2.3.4 Coriolis Effect

The Coriolis effect models the effects of the inertial reference frame being affixed to a rotating Earth. This manifests itself as extra forces and moments that act on the vehicle. Since this term, like added mass, depends on the vehicle configuration, it will be neglected for this thesis, as in [45].

### *2.3.5 Thruster Model*

Although an actual vehicle is not available for testing, a simple thruster model, along with their configuration on the vehicle, is needed to properly design the controller. From Caccia *et al.* [45], most thruster systems have a negligible time constant when compared to that of the vehicle. Additionally, the drag effects of the thruster are indirectly accounted for by the drag term in the vehicle dynamic model (2.26). With this in mind, a single thruster's force is modeled as:

$$T = k_p T_{\text{max}}, \quad -1 \leq k_p \leq 1 \tag{2.28}$$

where $k_p$ is the percentage of power supplied to the thruster and $T_{\text{max}}$ is the maximum (or saturation) force supplied by the thruster (see Figure 2.8).

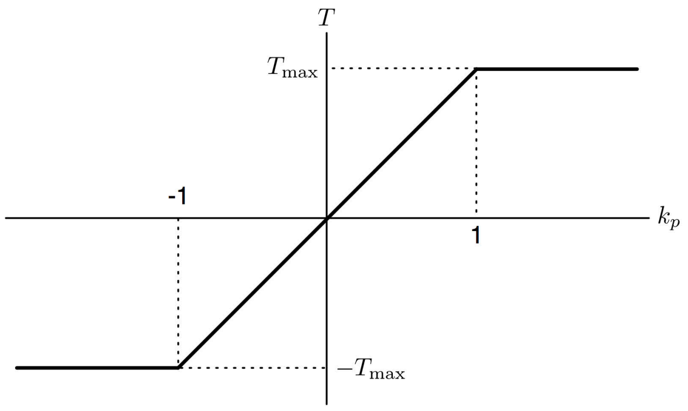


**Figure 2.8:** Thruster model.

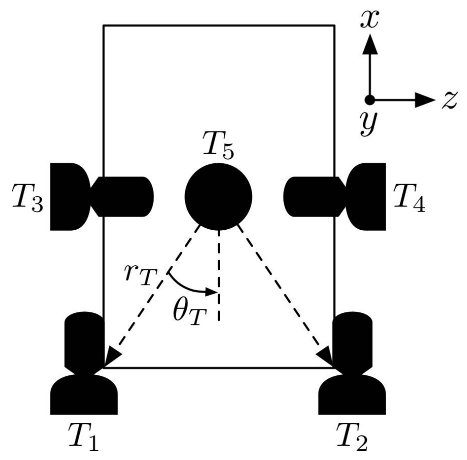


**Figure 2.9:** The chosen thruster layout.

With the absence of an actual vehicle, a thruster configuration based on [13], [21] and the Seamor 300T [4] is chosen as shown in Figure 2.9. Using this layout, the total thruster force/moment vector is calculated as:

$$\boldsymbol{F}_T(t) = \boldsymbol{\Gamma}_T \boldsymbol{T} \tag{2.29}$$

where $\boldsymbol{T} = \begin{bmatrix} T_1 & T_2 & T_3 & T_4 & T_5 \end{bmatrix}^T$ is a vector containing the forces of each thruster and $\boldsymbol{\Gamma}_T$ is a mapping of forces to the body-fixed frame defined as (for the given configuration):

$$\boldsymbol{\Gamma}_T = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -r_T \sin(\theta_T) & r_T \sin(\theta_T) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.30}$$

This mapping of thruster forces based on configuration is similar to that discussed by Choi and Kondo [12]. The interested reader may refer to this research concerning fault-tolerant control in the case of one or many thrusters failing. Improvements could then be made to the thruster configuration and control scheme.

## *2.3.6 Simplified Model*

With the simplifications and approximations discussed above, the simplified TUV model is written as:

$$\mathrm{M}\ddot{\boldsymbol{\zeta}} = \boldsymbol{F}_B(\boldsymbol{\eta}) + \boldsymbol{F}_D(\dot{\boldsymbol{\zeta}}_{\text{rel}}) + \boldsymbol{u}(t) \tag{2.31}$$

Expanding along one of the linear DOF (surge, sway, or heave), the buoyancy term drops out and the dynamic equation becomes:

$$m\ddot{\zeta} = -k_\zeta \dot{\zeta}_{\text{rel}} - k_{\zeta|\zeta|}\dot{\zeta}_{\text{rel}} \left|\dot{\zeta}_{\text{rel}}\right| + u(t) \tag{2.32}$$

From this, the model parameters that need to be identified are $m$, $k_\zeta$ and $k_{\zeta|\zeta|}$ for each DOF. Mass and inertia are easy to measure or calculate for each DOF; the drag coefficients, however, need to be determined experimentally. This can be done through tow tank testing [46] or various online parameter identification methods [21, 45].

## 2.4 TUV Control Continuum

With a suitable vehicle model outlined, a controller can be developed to meet the goals defined in §1.2. First, a method must be developed through which the operator has control over the vehicle over a wide range of operating ranges and, therefore, time delays. In order for existing ROV operators to require as little training as possible, it is preferable to maximize the range that supports some form of close control of the vehicle. As discussed in §1.2, no current control paradigms can easily do this for a generic TUV system without manual reconfiguration. This thesis proposes a new control paradigm that creates a smooth evolution through existing control schemes, loosely based on the measured time delay, forming a continuum of control. The logical evolution through the continuum is illustrated in Figure 2.10. At small time delays, the continuum starts in close control only. As the control evolves, a higher dependence on autonomy is gradually placed on the vehicle. The variables in the figure, $t_{c/p}$, $t_{p/s}$, and $t_{s/a}$, represent key times in the continuum and will be discussed later.

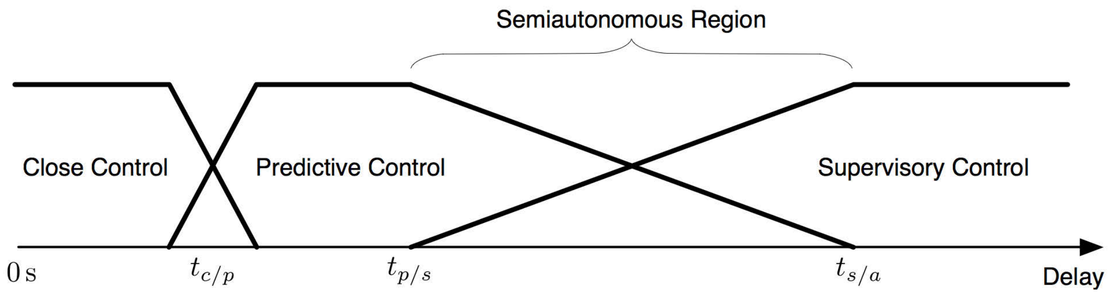


**Figure 2.10:** The logical evolution through the control continuum based on the time delay (based on a figure in [47]).

In order to achieve the desired continuum evolution, a number of existing techniques are blended in a novel manner. The overall block diagram of this combination is shown in Figure 2.11. It is similar to the predictor display shown in Figure 2.5, but with additional "smarts". With reference to the block diagram, the operator is responsible for tracking the overall desired state, $\boldsymbol{x}_d$, given output from the predictor display as well as the actual vehicle. The commands generated by the operator, $\boldsymbol{T}_{\text{op}}$, are sent to both the predictor display and the remote vehicle. The predictor uses these commands, in addition to the delayed vehicle state, $\boldsymbol{x}$, to create a prediction, $\boldsymbol{x}^*$, which is fed to both the operator and the vehicle. The vehicle uses the prediction as a waypoint for its on-board autpilot system

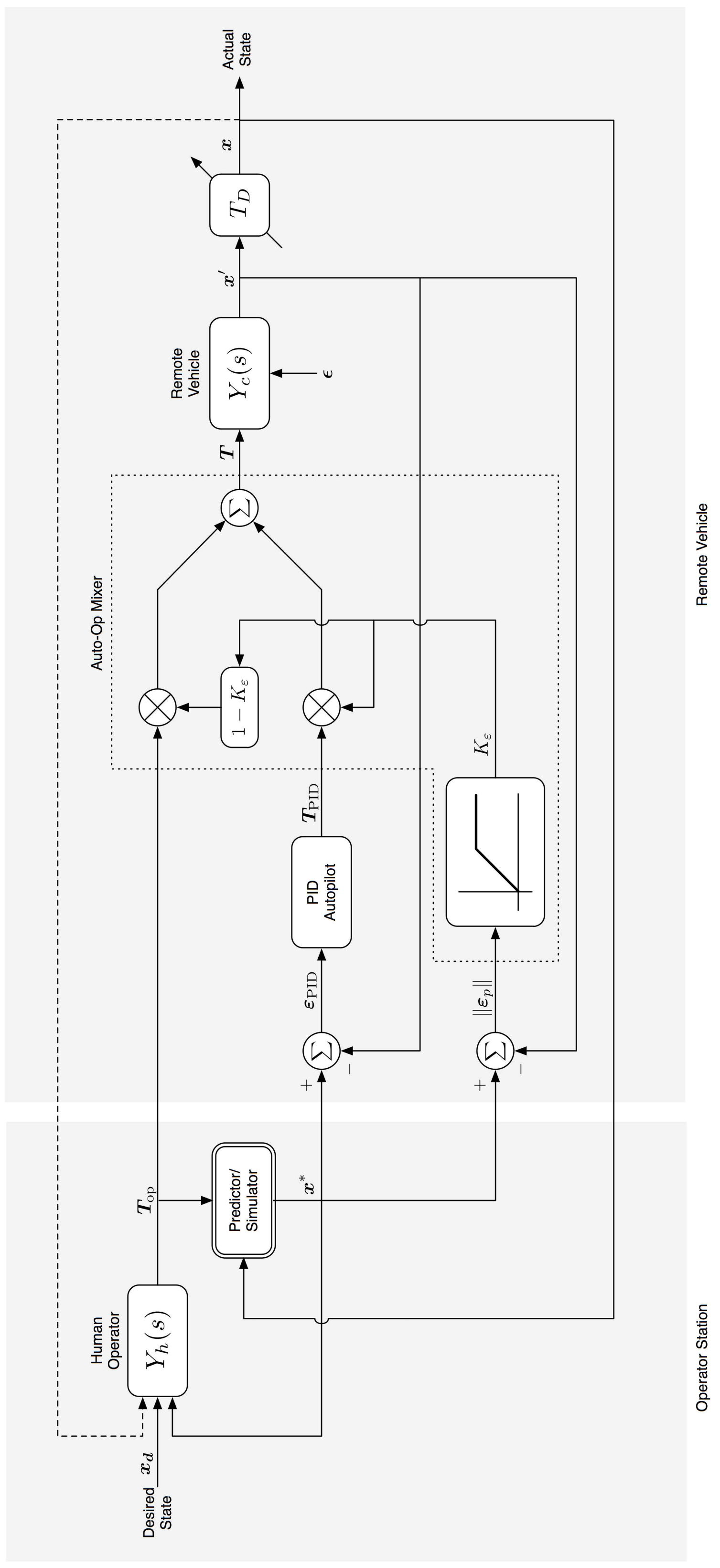


**Figure 2.11:** Block diagram of the control continuum.

that generates command $\boldsymbol{T}_{\mathrm{PID}}$. The key contribution of this research, the autopilot-operator command mixer (referred to as the "auto-op mixer"), is then used to create the actual command issued to the thrusters, $\boldsymbol{T}$.

Figure 2.11 also illustrates how the control system is distributed across the operator station and the remote vehicle. That is, the operator station is responsible for the predictor system, while the remote vehicle is responsible for the autopilot calculations. The advantage of having the autopilot system on the remote vehicle is that it allows the controller to deal with large communication drop-outs (problems with the acoustic link). Since the vehicle stores a list of predictions (waypoints) that have already been sent by the operator station, if a large drop-out occurs, the autopilot can completely take over and use the stored waypoints until the link is re-established. Similarly at the operator station, the prediction system can be turned off in the absence of measurements from the remote vehicle until more are available. When communications are re-established, the auto-op mixer can be used as-is to correct for any large discrepancies. The following subsections analyze each component of the continuum control loop in more detail.

### *2.4.1 Close Control Phase*

Close control is the lowest level of control in the continuum. In this mode, the operator has unmodified and almost instantaneous control of the vehicle. This is comparable to tethered ROV operation due to the high acoustic bandwidth and very short time delay afforded at short ranges, and so, existing ROV pilots would require little to no additional training. A control station is required to accommodate the operator's interaction with the vehicle, so a simple interface is used for this thesis (a mockup of which is shown in Figure 2.12).

Close control is intended for tasks that require frequent and precise interactions between the operator and remote vehicle such as worksite inspection, wreck investigation, filmmaking, etc [47]. As such, the limiting factor in this mode is the time delay, since human control tends to destabilize between $1/3$ s and $1/2$ s (as discussed in §2.1.1 and indicated in Figure 2.10 as $t_{c/p}$). This limits the range of unaided, close control to anywhere from 250 to 400 m. At this point, the continuum starts evolving in order to aid the operator and maintain the illusion of close control.

### *2.4.2 Predictive Phase*

Predictive displays are an already well-established control paradigm, especially in the planetary rover industry [44] where they are used to alleviate the several-second control

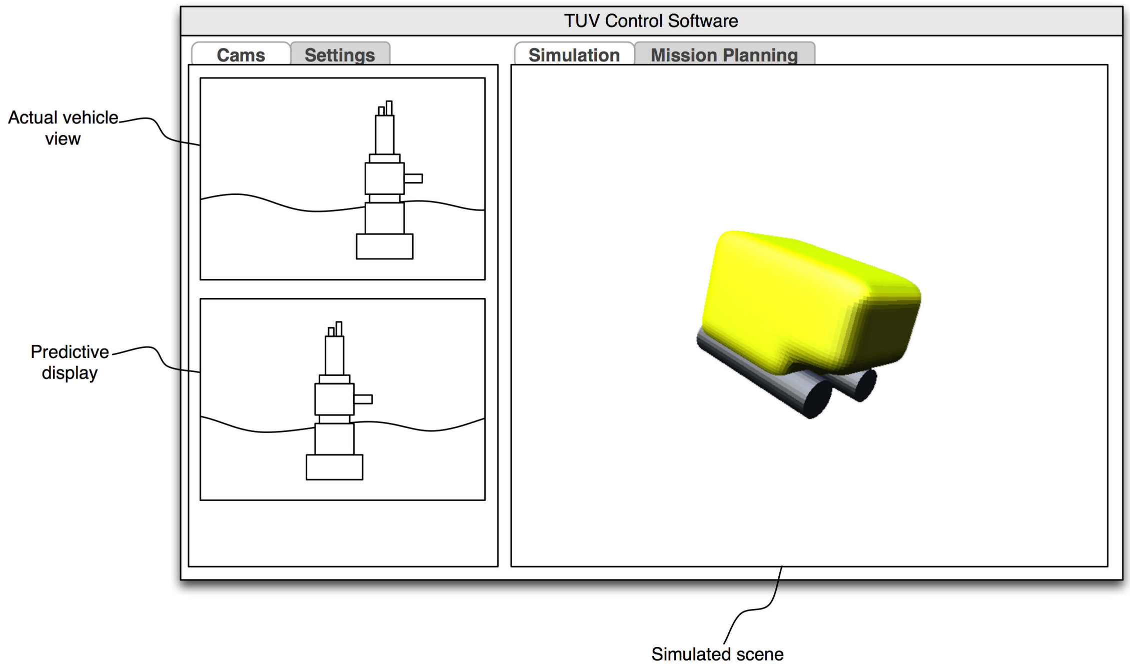


**Figure 2.12:** Concept for operator station interface, presenting only the required information to test the control continuum.

delay. This application, however, does not require the controller to cope with the varying delay that is caused by the acoustic link as with TUVs. To deal with this, the controller in this research keeps a history of all inputs issued by the operator for a period of time equal to the time delay [47], referred to as a “command queue”. During each prediction stage, any “stale” commands (commands that are older than the time delay) are removed from the queue since they have already been received by the remote vehicle and incorporated in the state measurement. The remaining commands are then integrated up to the current time with the fast-time prediction method. The concept can be visualized as a moving integration window over the command queue whose width depends on the time delay, $T_D$, as in Figure 2.13.

The fast-time prediction utilizes the Euler method to solve the simplified dynamic model, (2.31), in all 6 DOF. The Euler method is an application of the first two terms of the Taylor series expansion for solving ordinary differential equations (ODEs), given an initial value. The Euler method, starting from time $t_0$ to time $t_0 + h$ is written as:

$$y(t_0 + h) = y(t_0) + h\, y'(t_0) \tag{2.33}$$

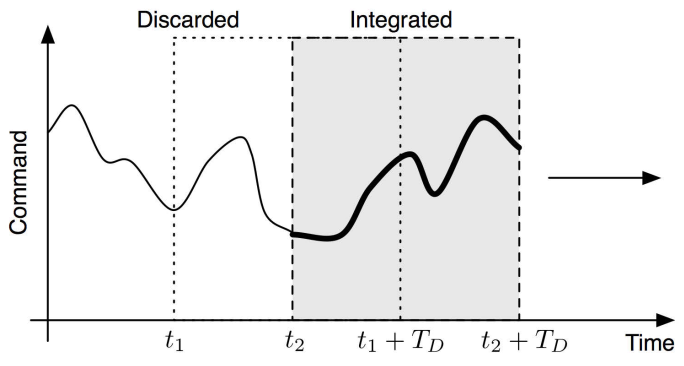


**Figure 2.13:** Command queue with the moving integration window. The stale commands are discarded from the queue after they are received by the vehicle (based on a figure in [47]).

For this to work, the Euler time step, $h$, is assumed to be much less than the vehicle response time. To apply the Euler method to our system in one DOF, (2.32), we must first reduce the equation to a pair of first-order ODEs. First, let $y_1 = \zeta$ and $y_2 = \dot{\zeta}$, then the system can be represented as:

$$\begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = \frac{1}{m}\left(-k_\zeta y_2 - k_{\zeta|\zeta|} y_2 \left|y_2\right| + u\right) \end{cases} \tag{2.34}$$

Applying the Euler method (2.33) to the system (2.34) we have:

$$\begin{cases} y_1(t_i + h) = y_1(t_i) + h\, y_2(t_i) \\ y_2(t_i + h) = y_2(t_i) + h\, \frac{1}{m}\left(-k_\zeta y_2(t_i) - k_{\zeta|\zeta|} y_2(t_i) \left|y_2(t_i)\right| + u(t_i)\right) \end{cases} \tag{2.35}$$

where $t_i$ is the initial time of the current fast-time prediction step and $u(t_i)$ is the operator command at time $t_i$, saved in the command queue. Each prediction is recursively used as the initial state for the next time step until the entire time delay, $T_D$, has been predicted. The error associated with this method is proportional to the time step, $h$, so it is desirable to have it as small as computationally possible, since reducing $h$ increases the number of predictions required for the entire delay. This also allows the incorporation of more operator commands into the prediction span. The above simultaneous equations can be combined into one state vector, $\boldsymbol{y} = \begin{bmatrix} y_1 & y_2 \end{bmatrix}^T$, representing the position and velocity in the current DOF. It should also be noted that since this prediction deals with coordinates in the body-fixed frame, the final predicted state is relative to the initial state and may need to be transformed to the inertial reference frame.

The resulting fast-time prediction is then used to update a real-time simulation of the vehicle in its environment. The simulated scene is combined with the delayed information from the actual robot and presented to the operator. This is done through the use of a "ghost" robot that is overlaid on a virtual environment constructed using the delayed state measurements, similar to the methods used in [32, 48] (see Figure 2.14). This method allows the operator to control the ghost vehicle in real-time as if there was no communications time delay, maintaining the illusion of close control. The commands issued by the operator to the ghost are sent to the remote vehicle and also saved to the command queue for later prediction. Every time a measurement is received from the remote vehicle, the simulation is immediately updated to reflect the new prediction. This can create sudden jumps in the position of the ghost vehicle if the correction is large. For this reason, a threshold, $\varepsilon_c^{\max}$, is placed on the correction distance, $||\boldsymbol{\varepsilon_c}||$. If the correction distance is beyond this threshold, the correction will not occur and the prediction will be discarded for that step (this will be discussed more in the following section).

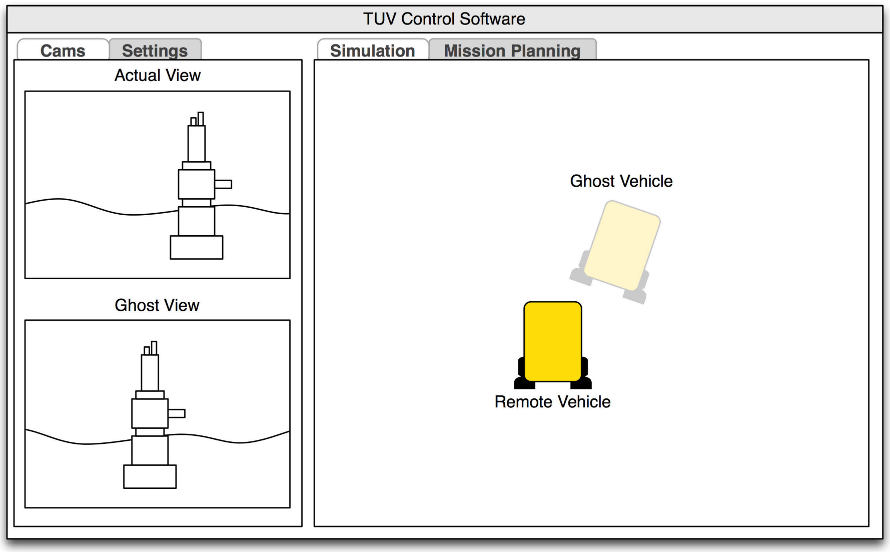
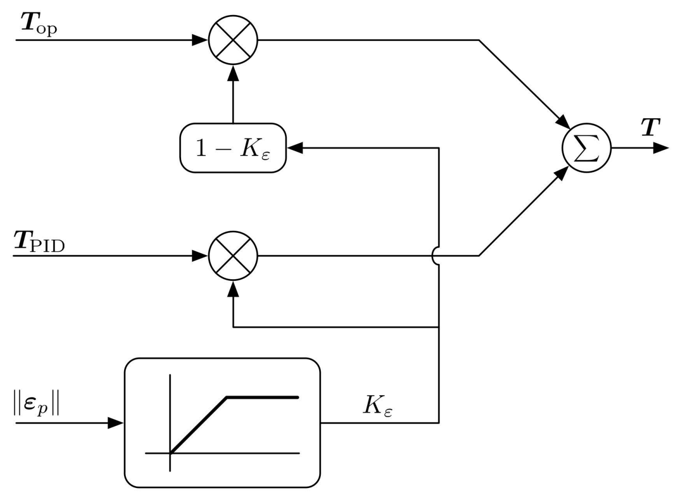


**Figure 2.14:** Ghost overlay on the operator station.

The limiting factor in this mode is also, indirectly, the time delay. As the time delay increases, the prediction span must also increase which increases the cumulative prediction error. Beyond a certain time, $t_{s/a}$ in Figure 2.10, the prediction error will always be over the threshold, making the fast-time prediction useless beyond this point. The time $t_{s/a}$

depends on two factors: the operator's prediction error tolerance, and the Euler prediction time step, $h$, afforded by the processing power.

### *2.4.3 Semiautonomous Region*



$\boldsymbol{T}_{\text{op}}$
$1 - K_\varepsilon$
$\boldsymbol{T}_{\text{PID}}$
$\|\boldsymbol{\varepsilon}_p\|$
$K_\varepsilon$
$\sum$
$\boldsymbol{T}$

**Figure 2.15:** The auto-op mixer component extracted from Figure 2.11 (based on a figure in [47]).

As mentioned, the prediction error increases with increasing time delay. With large prediction errors, the commands issued by the operator corresponding to the predicted ghost vehicle may not be valid for the remote vehicle. For this reason, this research included a semiautonous region in the control evolution. This is realized through a command mixer as shown in Figure 2.15 and written as:

$$\boldsymbol{T} = K_\varepsilon \boldsymbol{T}_{\text{PID}} + (1 - K_\varepsilon)\boldsymbol{T}_{\text{op}} \tag{2.36}$$

Here, the commands generated by the operator, $\boldsymbol{T}_{\text{op}}$, are mixed with commands generated by the autopilot, $\boldsymbol{T}_{\text{PID}}$, through a mixing factor, $K_\varepsilon$ (referred to as the "auto-op mixing factor"). This produces an augmented thruster command, $\boldsymbol{T}$, that is used to control the actual vehicle. The mixing factor is determined by the error between the current state of the vehicle ($\boldsymbol{x}'$ in Figure 2.11) and the state predicted from the delayed measurements ($\boldsymbol{x}^*$), according to the relationship as shown in Figure 2.16. This allows the auto-op mixer to respond to instantaneous increases in prediction error and also the increase in error associated with the increasing time delay. The auto-op mixing parameters, $\varepsilon_p^{\text{low}}$ and $\varepsilon_p^{\text{high}}$, are chosen based on the operator's error tolerance.

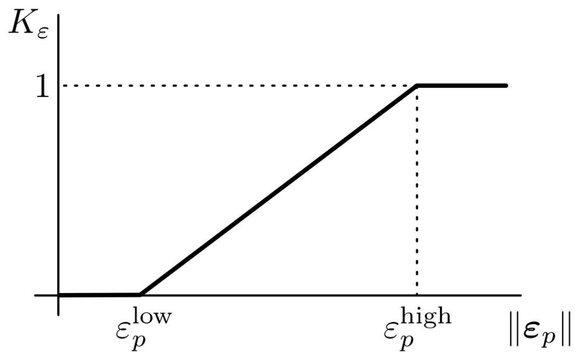


**Figure 2.16:** Auto-op mixing factor vs. prediction error magnitude.

The main goal of the auto-op mixer is to increase the positioning accuracy of the remote vehicle to better reflect what the operator sees with the ghost vehicle. If the prediction is accurate (i.e. the predicted ghost vehicle's state is close to the actual vehicle's state), then the commands issued by the operator to accomplish a goal in the predicted scene may be applied to the actual vehicle. Here, the mixer does not need to modify the commands and the mixing factor, $K_\varepsilon$, is close to 0. If, however, the prediction is erroneous, then commands issued by the operator in the simulated environment cannot be used to control the actual vehicle and the auto-op mixer must augment them with those of the autopilot (so $K_\varepsilon$ is close to 1).

The maximum range of the semiautonomous region coincides with the effective range of the predictor. That is, when the predictions associated with a certain time delay, $t_{s/a}$ in Figure 2.10, produce prediction errors consistently over the threshold, $\varepsilon_p^{\text{high}}$, then the mixer is no longer needed. Conversely, the starting delay time of the auto-op mixer, $t_{p/s}$, is the delay corresponding to the lower error bound, $\varepsilon_p^{\text{low}}$, and the accuracy of the predictor implemented.

### *2.4.4 Supervisory Phase*

When the fast-time predictions are no longer adequate, the continuum evolves into a supervisory phase. Here, the operator does not directly control the vehicle, but instead only creates waypoints (denoted $\boldsymbol{x}^*$ in Figure 2.11) that are sent to the remote vehicle. These waypoints can be constructed in one of two ways [47]:

1. By the operator controlling the ghost simulation in real-time (or possibly a time-lapse version where it moves faster than reality [44]). From the motions of the ghost, waypoints are created by the controller and sent to the vehicle.

2. By editing waypoints directly in a mission planning view (like many AUV systems) giving the operator precise control over each waypoint [8].

An on-board autopilot is used to track the waypoints automatically, while the operator can monitor the (delayed) progress in the control software (as illustrated in Figure 2.17). In order for the remote vehicle to properly mirror the desired motions of the operator, the waypoints require both position and orientation information. A waypoint vector is constructed with the same form as the inertial-frame pose vector (2.13), but without the roll and pitch angles (since these are passively controlled):

$$\boldsymbol{x}^* = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^T \tag{2.37}$$

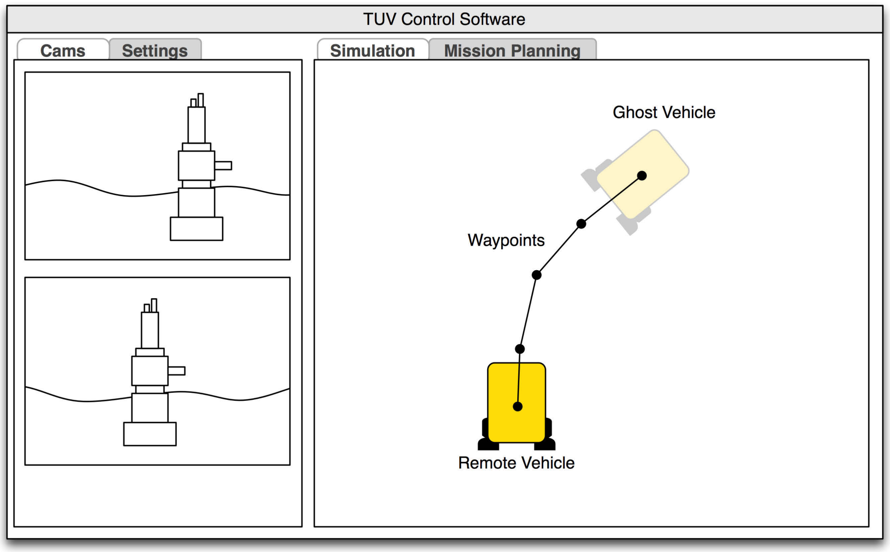


**Figure 2.17:** Waypoints generated by the ghost vehicle and sent to the actual vehicle.

The frequency at which the waypoints are sent to the vehicle also depends on the time delay. With larger delays, and therefore operating distances, the acoustic bandwidth is reduced. Using the data rate-range product defined in §1.1.2, the waypoints should be sent at a rate $\geq 10\,{}^{ms}\!/\mathrm{km}$, to ensure each waypoint is received and processed before the following one. Since the operator is only interacting with the vehicle through waypoints, the maximum range of supervisory mode is determined by the maximum range of the acoustic link employed. As mentioned before, some research suggests ranges of up to 200 km [20].

**PID Autopilot**

The type of autopilot implemented for this thesis is a PID controller, due to the robustness and ease of implementation. PID controllers have already been proven to be an effective autopilot controller for underwater vehicles [21, 23, 30]. Following Steinke's approach [21], a PID configuration is used for positioning the vehicle, while PD is used for rotation. It is also assumed that depth operations are decoupled from motions in the XZ-plane, so depth is controlled independently with its own PID controller. In the XZ-plane, the PID controller tracks the position error vector, $\Delta\boldsymbol{r} = \begin{bmatrix} \Delta r_x & \Delta r_z \end{bmatrix}^T$, and the heading error, $\Delta\psi$. Roll and pitch are not controlled by the PID controller since they are passively controlled by the buoyancy. These errors can be combined into a single PID error vector for the XZ-plane:

$$\boldsymbol{\varepsilon}_{\text{PID}}^{x/z} = \begin{bmatrix} \Delta\psi & \Delta\dot{\psi} & \Delta r_x & \int\Delta r_x dt & \Delta\dot{r}_x & \Delta r_z & \int\Delta r_z dt & \Delta\dot{r}_z \end{bmatrix}^T \tag{2.38}$$

Similarly, for depth:

$$\boldsymbol{\varepsilon}_{\text{PID}}^{y} = \begin{bmatrix} \Delta r_y & \int\Delta r_y dt & \Delta\dot{r}_y \end{bmatrix}^T \tag{2.39}$$

Then, with respect to the thruster mapping outlined in §2.3.5, the PID controls each thruster force in the XZ-plane according to:

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} -K_\psi^P & -K_\psi^D & K_x^P & K_x^I & K_x^D & 0 & 0 & 0 \\ K_\psi^P & K_\psi^D & K_x^P & K_x^I & K_x^D & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & K_z^P & K_z^I & K_z^D \\ 0 & 0 & 0 & 0 & 0 & -K_z^P & -K_z^I & -K_z^D \end{bmatrix} \boldsymbol{\varepsilon}_{\text{PID}}^{x/z} \tag{2.40}$$

and for depth:

$$T_5 = \begin{bmatrix} K_y^P & K_y^I & K_y^D \end{bmatrix} \boldsymbol{\varepsilon}_{\text{PID}}^{y} \tag{2.41}$$

Where the $K$'s are the gains for the PID controller that must be tuned for each DOF. Both of the above equations are combined into a single, commanded thrust vector from the PID controller:

$$\boldsymbol{T}_{\text{PID}} = \begin{bmatrix} T_1 & T_2 & T_3 & T_4 & T_5 \end{bmatrix}^T \tag{2.42}$$

**Thruster Saturation**

One problem is that the PID controller has no knowledge that thrusters can only provide the maximum force $\pm T_{\text{max}}$. If one of the errors is large, the proportional gain term, $K^P$, may command a thruster force that is far beyond the capabilities of the thrusters. In order to maintain the desired control action, the thruster command force must be clamped to the maximum value. First, the maximum value of the thrust vector is calculated:

$$T_{\text{PID}}^{\text{max}} = \max\left\{|\boldsymbol{T}_{\text{PID}}|\right\} \tag{2.43}$$

Where $|\cdot|$ here, represents the absolute value of each vector component and not the vector norm. Then, if this value exceeds the maximum capable thrust, $T_{\text{max}}$, then the clamped vector is calculated as:

$$\boldsymbol{T}_{\text{PID}}{}' = \begin{cases} \frac{T_{\text{max}}}{T_{\text{PID}}^{\text{max}}}\boldsymbol{T}_{\text{PID}}, & T_{\text{PID}}^{\text{max}} > T_{\text{max}} \\ \boldsymbol{T}_{\text{PID}}, & \text{otherwise} \end{cases} \tag{2.44}$$

which limits all of the commanded values to realistic values.

Clamping the PID output introduces another problem called integrator windup [29]. When a thruster saturates, it effectively breaks the PID feedback loop since the thruster stays at its saturation value regardless of the PID output. When this happens, the integral term continuously integrates the error because the PID controller cannot minimize it fast enough. As a result, the integral term becomes very large, or “winds-up”, which creates large overshoot in the system output until the integral term is reduced by the opposite error value.

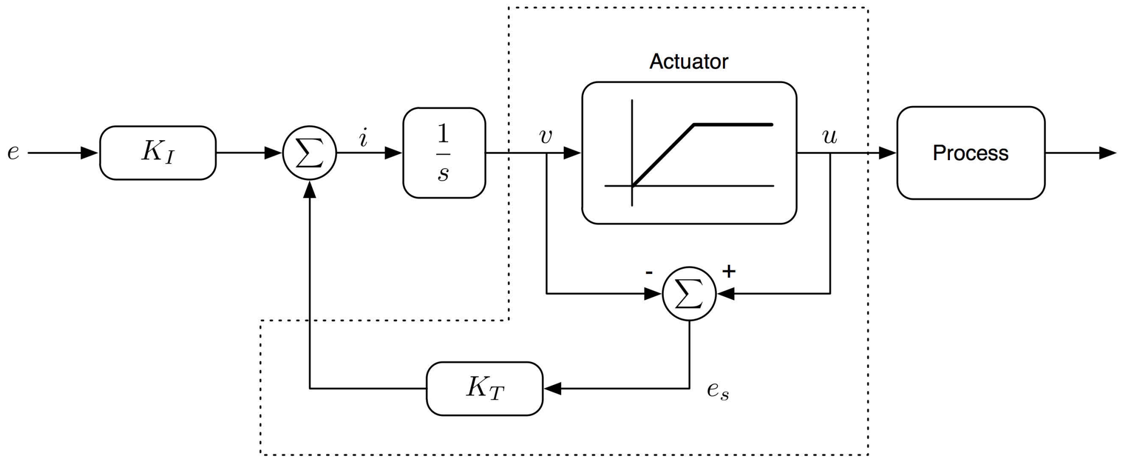


**Figure 2.18:** Integral term of a PID controller with an anti-windup compensator (based on a figure in [29]).

To eliminate this problem, an anti-windup compensator term needs to be added to the normal PID loop (as in Figure 2.18). It is in the form of an extra feedback loop constructed from the actuator error, $e_s = u - v$, which is the difference between the commanded and actual thruster output. This is fed back through compensator gain $K_T$, and added to the integrator input, $i$. During normal operation (i.e. thruster is not saturated), the compensator has no effect since $e_s = 0$. When the thruster saturates, the compensator is used to reduce the integral term preventing it from growing unbounded. This can be shown by analyzing the expression for the integral term:

$$i = K_I e + K_T e_s \tag{2.45}$$

In steady-state, after the thruster saturates $e_s = u_{\text{max}} - v$, and $i = 0$, so the commanded thruster force is written as:

$$v = u_{\text{max}} + \frac{K_I}{K_T} e \tag{2.46}$$

This indicates that the steady-state commanded force is only slightly different than the thruster saturation value. Once $e_s = 0$ again, there is no accumulated error so the integral term can once again be used as normal. From [29], a rule-of-thumb is to choose the compensator gain as $K_T = \sqrt{\frac{K_I}{K_D}}$.

# CHAPTER 3

# TUV SIMULATION DESIGN AND IMPLEMENTATION

With the controller designed, it is important to test the theory through practical use. In the absence of an actual test vehicle, this thesis relies on computer simulations to verify and test the proposed controller. The simulation developed in this chapter is meant to reproduce the scenario of flying a small underwater vehicle for worksite inspection or filmmaking tasks. As such, it will be assumed that the vehicle does not require actuated manipulators and instead, the operator is only interested in navigation and visual inspection of objects. That said, the simulation could be easily adapted to facilitate a specific vehicle configuration that includes the use of manipulators. This chapter first outlines the assumptions required in order to construct a generic simulation. Next, each component of the control continuum is tested individually to verify that they work as intended. Finally, the development of the complete simulation is discussed and the continuum implemented in full. The simulator is later used to carry out a simple navigation task to test the continuum, as is discussed in §4.3.

## 3.1 Simulation Assumptions

With no actual vehicle to directly model the simulation after, some typical values for a small vehicle must be chosen to allow the simulation to be useful. In terms of the vehicle model, the values that must be chosen are the mass and inertia matrix, $\mathbf{M}$, and the hydrodynamic drag coefficients, $k_\zeta$ and $k_{\zeta|\zeta|}$, for each DOF. Using [31, 45] as reference,

**Table 3.1:** Vehicle model parameters.

| | $\mathbf{M}$ | $k_\zeta$ | $k_{\zeta\|\zeta\|}$ |
|---|---|---|---|
| Surge | | | |
| Sway | $85\,\mathrm{kg}$ | $40\,\mathrm{N}^{s}/_{\mathrm{m}}$ | $200\,\mathrm{N}^{s^2}/_{\mathrm{m}^2}$ |
| Heave | | | |
| Pitch | $8\,\mathrm{kg\,m}^2$ | | |
| Yaw | $15\,\mathrm{kg\,m}^2$ | $20\,\mathrm{N\,s\,m}$ | $50\,\mathrm{N\,s\,m}$ |
| Roll | $15\,\mathrm{kg\,m}^2$ | | |

reasonable parameter values are chosen as shown in Table 3.1 for a small vehicle. In addition, the saturation force of the thrusters is chosen as $T_{\text{max}} = 50\,\mathrm{N}$ (M. Rowsome, DRDC Atlantic, private communication. Sept 2010).

Apart from the vehicle model, assumptions need to be made about the vehicle and the environment. First, a map of the area of operations must be known and available to the simulation and the vehicle. This is required to build a simulated scene so that the operator can observe the position of the ghost vehicle relative to any objects of interest. Next, the vehicle must also be able to localize itself on this map and send this telemetry to the operator station. As discussed before, there has been previous work using terrain matching and computer vision techniques to accomplish this with high accuracy [23–25], as well as the standard underwater vehicle positioning systems (such as USBL). Finally, it must be assumed that the navigation and video telemetry can be sent through the acoustic link reliably to the operator station. With current acoustic modem research, the performance of acoustic links is increasing, though they are still the limiting factor for this research.

## 3.2 MATLAB Proof-of-Concept

Before testing the complete control system and constructing a full simulation, each component of the controller is tested individually with MATLAB. This allows for the identification of any fundamental problems and also facilities tuning model parameter values. To do this, each component from the complete control loop (Figure 2.11) is extracted and implemented in MATLAB. Since the motion is decoupled in all DOFs, only a single DOF is tested at any given time. After each component is tested and verified, then it is ready to implement in the full simulator as discussed in §3.3.

### *3.2.1 Vehicle Model Verification*

To ensure the chosen model of the TUV (2.32) is accurate, it is instructive to observe the response to a step input. Using the model parameters in Table 3.1 and a step thruster force of 50 N, the response of the vehicle is shown in Figure 3.1. The response was calculated using the 4th-order Runge-Kutta (RK4) method. The RK4 method is especially useful for solving higher-order, nonlinear DEs, such as the vehicle model here. It is more computationally expensive than the Euler method, so if the Euler method can be shown to be sufficient, it is the preferable choice.

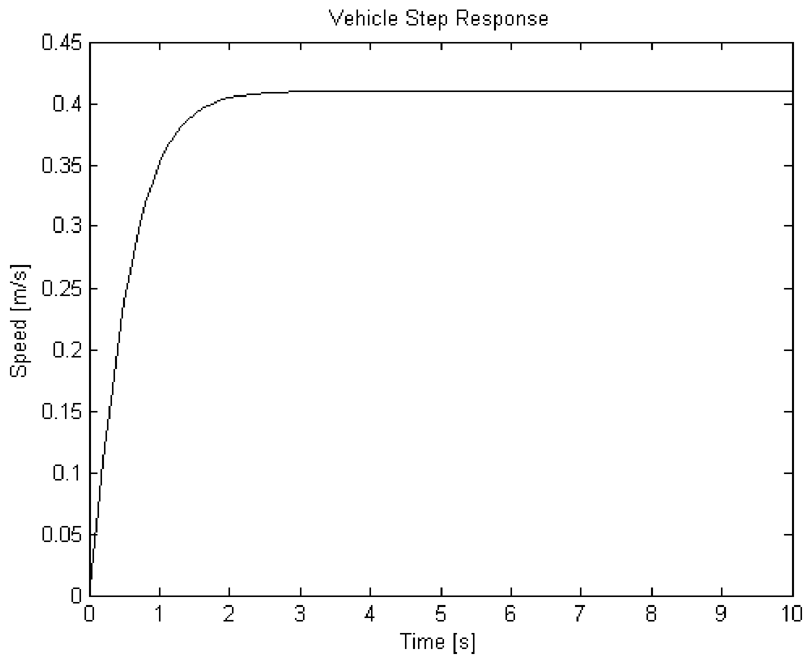


**Figure 3.1:** Step response of vehicle model given chosen parameter values.

From the graph, the maximum forward speed of the vehicle is approximately 0.4 m/s. This value agrees with typical maximum speed values for AUV and ROV systems. Also from the plot, the vehicle response time is in the order of 1 s. In order to verify the use of the Euler method for the prediction, the response generated by the Euler method should closely match that of the RK4. Choosing an Euler time step of $h = 0.01\,\mathrm{s}$, two orders of magnitude below the vehicle response, the response generated from the Euler method compared to the RK4 result is shown in Figure 3.2 (the MATLAB source used to create this is listed in Appendix A.1). This plot shows negligible error between the Euler and RK4 numerical methods. So, for this thesis the Euler method with a time step of 10 ms will be used to solve the system dynamic equations.

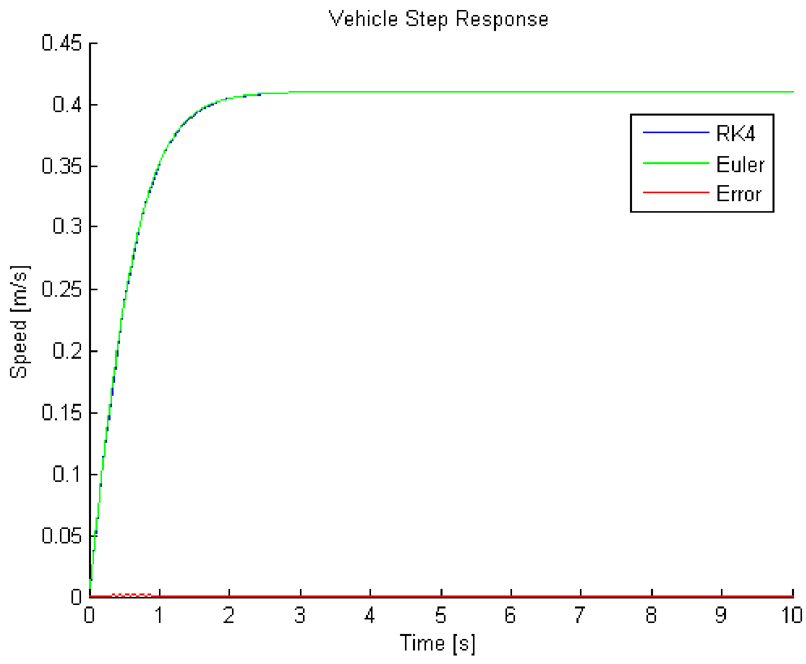


**Figure 3.2:** Comparison of Euler and RK4 numerical methods with a step thruster input.

### *3.2.2 PID Tuning*

With a suitable vehicle model implemented, the PID controller can now be tuned in place. To accomplish this, a simple 2D simulation is developed in MATLAB that utilizes the Euler method to solve the vehicle dynamic model for the surge and sway DOF. A PID controller is constructed, as outlined in §2.4.4, that calculates the thruster forces that are applied to the vehicle at each simulation step. The PID controller is fed a series of waypoints (PID set points) it must attempt to pilot the vehicle towards (i.e. minimize the error). To make the task easier, a waypoint is defined as a target area rather than a single point. Initially, the controller is constructed with no anti-windup compensator.

For the first test, a single waypoint is selected directly in front of the vehicle so only one DOF is actuated by the PID controller (see Figure 3.3). The waypoint position is

Vehicle Starting Position
Target Area

**Figure 3.3:** The layout of the PID tuning scenario with a single waypoint.

chosen far enough away from the vehicle to ensure it has reached steady-state (i.e. max speed) before reaching the goal. As a first attempt, the Ziegler-Nichols tuning method was applied to tune the PID gains. First, the proportional gain, $K_P$, is set to a value such that the vehicle oscillates around the waypoint. This corresponded to an ultimate gain value, $K_u$, of approximately 500, with an oscillation period, $P_u$, of 2 s (see Figure 3.4). Then, the PID gains are chosen as follows:

$$\begin{aligned} K_P &= 0.6K_u = 300 \\ K_I &= 2\frac{K_P}{P_u} = 300 \\ K_D &= \frac{K_P P_u}{8} = 75 \end{aligned}$$

However, when implementing these gains the overshoot is worsened, as shown in Figure 3.5. This is due to the effect of integrator windup, as discussed in §2.4.4. Since the thrusters saturate, the integral term of the PID controller grows too large, causing increased overshoot in the response. After implementing the anti-windup compensator and setting its gain to $K_T = \sqrt{\frac{K_I}{K_D}} = 2$, the improved PID response is shown in Figure 3.6. Here, the vehicle's performance is greatly improved over the controller with no compensator. This can be extended to a 2D simulation where the vehicle tracks multiple waypoints. The layout of this test is shown in Figure 3.7 with the order the vehicle must visit the waypoints marked in the figure as well as the desired path. Additionally, this test introduces a constant ocean current to test the PID's ability to cope with an unmodeled disturbance. Figure 3.8 shows the resulting simulated PID response.

As can be seen from the simulated response, the PID autopilot performs very well in the provided test case. The integral term is able to compensate for the ocean disturbance

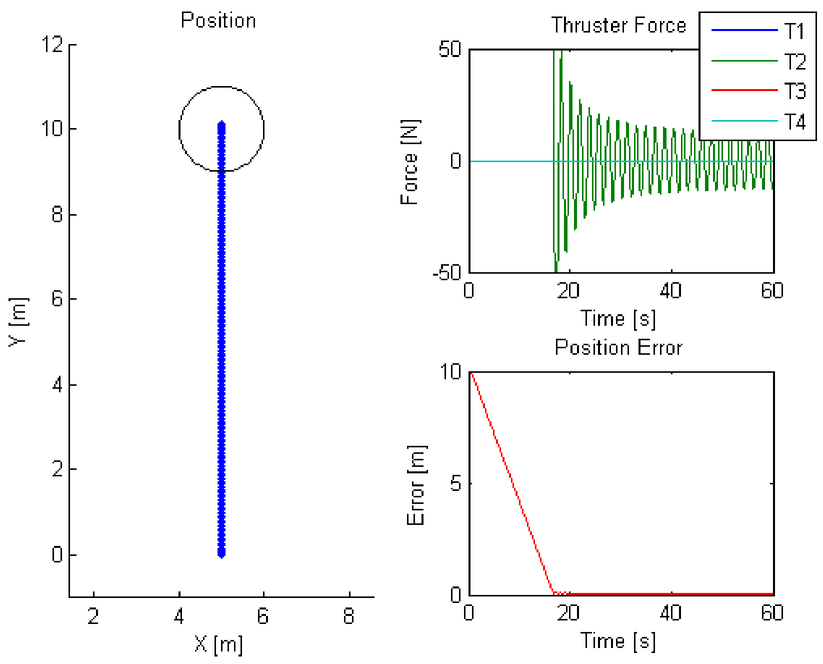


**Figure 3.4:** PID proportional gain, $K_P$, set to produce oscillations.

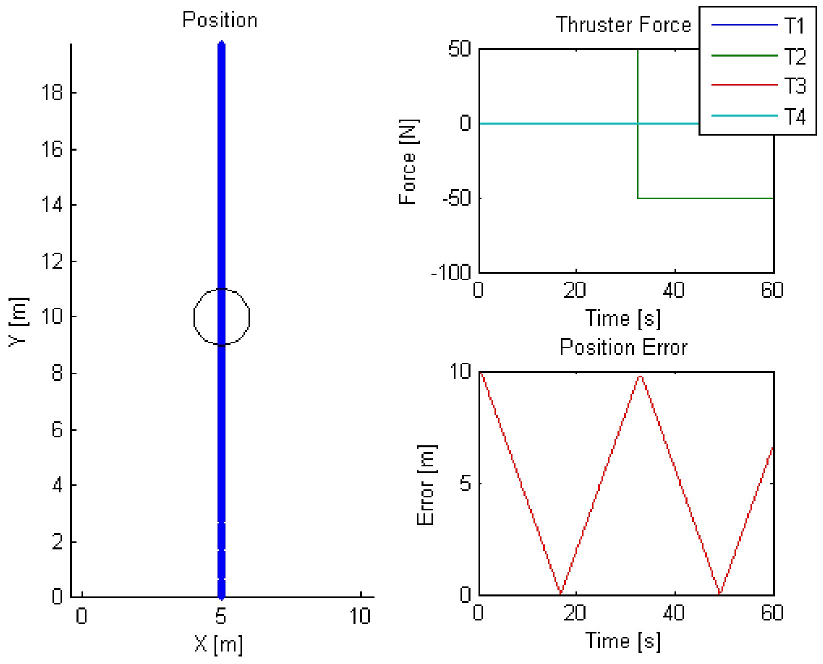


**Figure 3.5:** PID response with gains from the Ziegler-Nichols tuning method.

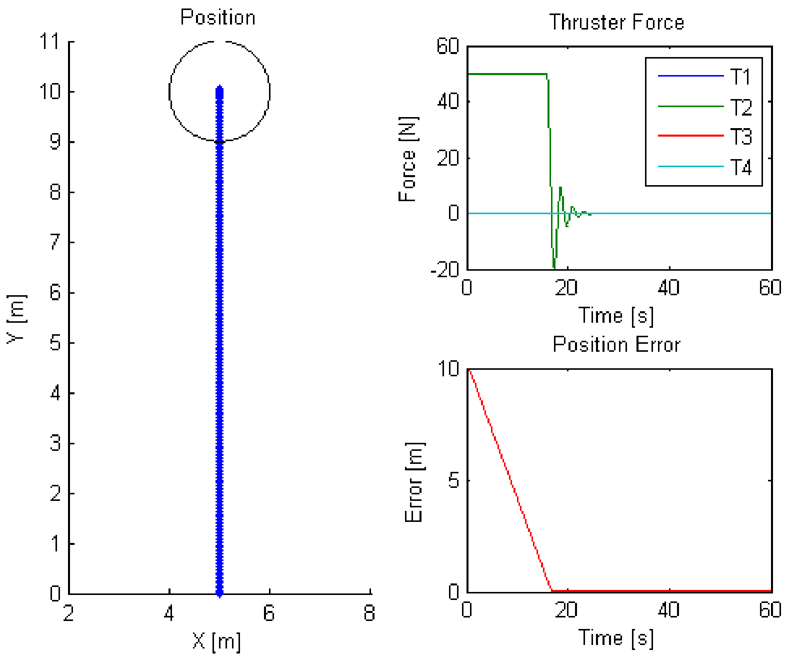


**Figure 3.6:** PID controller response with anti-windup compensator.

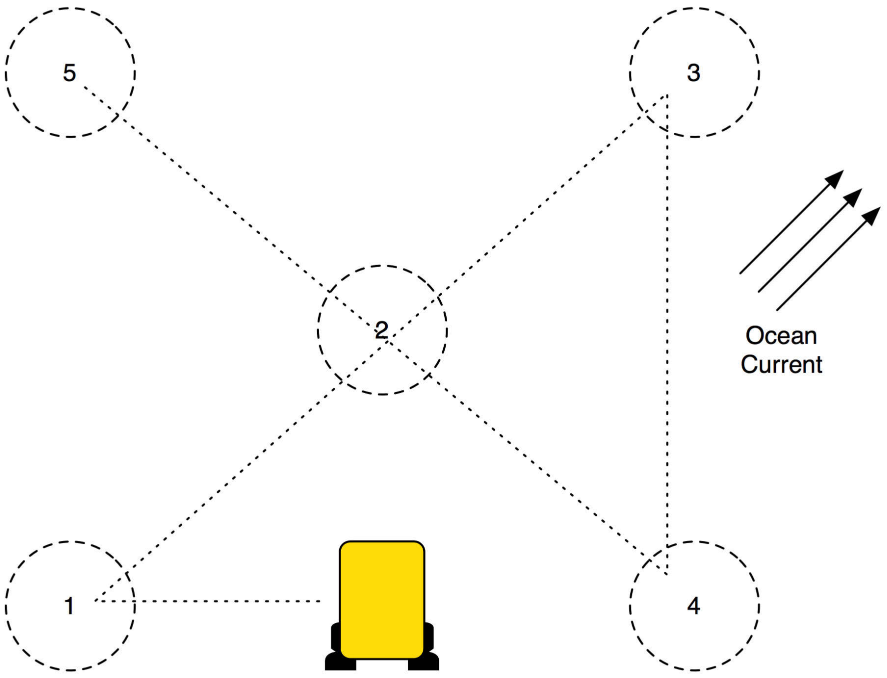


**Figure 3.7:** A test of the PID controller in 2-DOF with multiple waypoints.

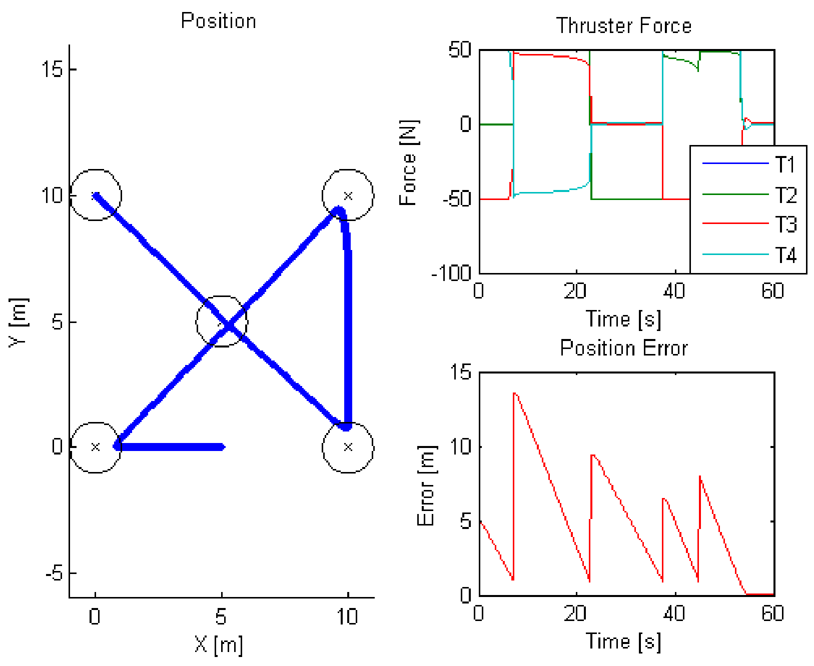


**Figure 3.8:** The simulated PID response of the test setup in Figure 3.7.

**Table 3.2:** Summary of chosen PID gains.

| Gain | Value |
|---|---|
| $K_{\psi}^{P}$ | 300 |
| $K_{\psi}^{D}$ | 200 |
| $K_x^P$, $K_y^P$, $K_z^P$ | 300 |
| $K_x^I$, $K_y^I$, $K_z^I$ | 100 |
| $K_x^D$, $K_y^D$, $K_z^D$ | 200 |

and each waypoint is visited with no overshoot. The autopilot also keeps the vehicle at its maximum speed as long as possible to ensure the task is completed as quickly as possible. A similar approach is used to tune the yaw and heave DOF as well. Table 3.2 lists all of the final PID gains necessary to construct the PID controller as described by (2.40) and (2.41). Some of the gains have been manually adjusted (by trial and error) to improve the performance of the controller. The MATLAB code used to tune the PID autopilot is listed in Appendix A.2.

### *3.2.3 Auto-Op Mixer Tuning*

The last component requiring tuning in the continuum is the auto-op mixer itself. Some of these values are largely subject to the preference of the operator given the current task. As such, these values will always be available to the operator to adjust on demand at all times throughout the mission. Since a command mixer similar to this has not been found in any literature, there is no formal tuning method for the parameters. This thesis attempts to present some basic guidelines for choosing the parameters that resulted in adequate simulator performance. The parameter values that must be identified are: the waypoint period, $T_{wp}$; the actual vehicle state measurement period, $T_{\text{meas}}$; the ghost correction threshold, $\varepsilon_c^{\text{max}}$; and the auto-op mixing parameters, $\varepsilon_p^{\text{low}}$ and $\varepsilon_p^{\text{high}}$.

In order for the autopilot-controlled motion to match the ghost vehicle's motion as closely as possible, the waypoints should be sent to the remote vehicle as quickly as the acoustic link allows. As mentioned earlier, modern acoustics links should handle waypoint transmission at a period of $T_{wp} = 10\,{}^{ms}\!/\text{km}$. With the period below the response time of the vehicle, the motion resulting from the autopilot tracking system should closely match that of the ghost vehicle controlled by the operator.

Next, the vehicle state measurement period, $T_{\text{meas}}$, must be chosen so the simulator can predict the future state of the remote vehicle. Another way to consider $T_{\text{meas}}$ is the delayed sampling period of the remote vehicle's state. If chosen too small, then the ghost vehicle is being constantly corrected and at large time delays the correction distance is also large (due to unmodeled disturbances and cumulative prediction error), making manual control almost impossible. Conversely, if the period is chosen to large, then the simulated scene may not present an accurate representation of the actual vehicle. A good rule of thumb that balanced both frequent updates and controllability was to choose the period to be equal to the time delay, $T_{\text{meas}} = T_D$. So, as the time delay and prediction error increase, the ghost correction frequency decreases, maintaining the controllability of the ghost while also presenting the operator with an updated ghost vehicle (only updated less often).

Related to this is the correction distance threshold, $\varepsilon_c^{\text{max}}$. This places a limit on the distance the ghost can be corrected using the predicted state. The operator can set this preference depending on the task they want to accomplish. For free flight through the water, they might want to set this to a small value or zero (i.e. the ghost does not correct itself with the prediction) and leave it up to the on-board autopilot to follow the ghost. For tasks requiring precise motions, the operator would want the threshold set to a high value so the ghost vehicle is as accurate as possible. However, higher thresholds permit the ghost vehicle to "jump" further when updates are received and large corrections are necessary. This can make controlling the vehicle difficult and in the worst case, the ghost and actual vehicle can oscillate exactly out of sync with one another. So, care must be taken when choosing the correction threshold to avoid any control instability within the given situation (in terms of the unmodeled disturbance combined with the time delay). For this thesis, the threshold was set to $\varepsilon_c^{\text{max}} = 0.5\,\text{m}$.

Finally, the mixer parameters, $\varepsilon_p^{\text{low}}$ and $\varepsilon_p^{\text{high}}$, must be tuned as well. The upper parameter, $\varepsilon_p^{\text{high}}$, is the value at which the prediction error (the difference between the predicted future state and the actual future state) causes the auto-op mixer to be in autopilot-only operation (refer to Figure 2.16). It should be chosen to be equal to the selected correction distance threshold, $\varepsilon_c^{\text{max}}$, since if the error is larger than this threshold, the autopilot must take over to correct the error. To maximize the effectiveness of the mixer, the lower mixer parameter, $\varepsilon_p^{\text{low}}$, should be set to 0. This means that even small amounts of error will produce some augmentation of the commands by the auto-op mixer which is useful for small, sporadic disturbances. Table 3.3 summarizes the auto-op mixer parameters as chosen above.

**Table 3.3:** Summary of auto-op mixer parameters.

| Parameter | Value |
|---|---|
| $T_{wp}$ | range $\cdot\ 10\,{}^{ms}/_{\text{km}}$ |
| $T_{\text{meas}}$ | $T_D$ |
| $\varepsilon_c^{\text{max}}$ | $0.5\,\text{m}$ |
| $\varepsilon_p^{\text{low}}$ | $0\,\text{m}$ |
| $\varepsilon_p^{\text{high}}$ | $\varepsilon_c^{\text{max}} = 0.5\,\text{m}$ |

## *3.2.4 Control Continuum Component Testing*

Before implementing the continuum in the full simulator environment, the predictive, semiautonomous and supervisory control modes were tested separately to ensure they operated as desired. With the tuned parameters as outlined in the previous sections, each component was tested in MATLAB in 1- and 2-DOF.

### Predictive Phase

The goal of testing the predictive phase is to ensure that the fast-time Euler method performs sufficiently well in predicting the future state of the vehicle. The tests also allow the determination of the maximum time delay before the predictions become unusable (i.e. the prediction error is too large). The tests were carried out in a single DOF using a simulation of the ghost and actual vehicles. No operator or autopilot is used to generate commands, so a random command sequence is fed to the simulation. The response of the "actual" vehicle, along with the command input sequence is shown in Figure 3.9 with $T_D = 0$.

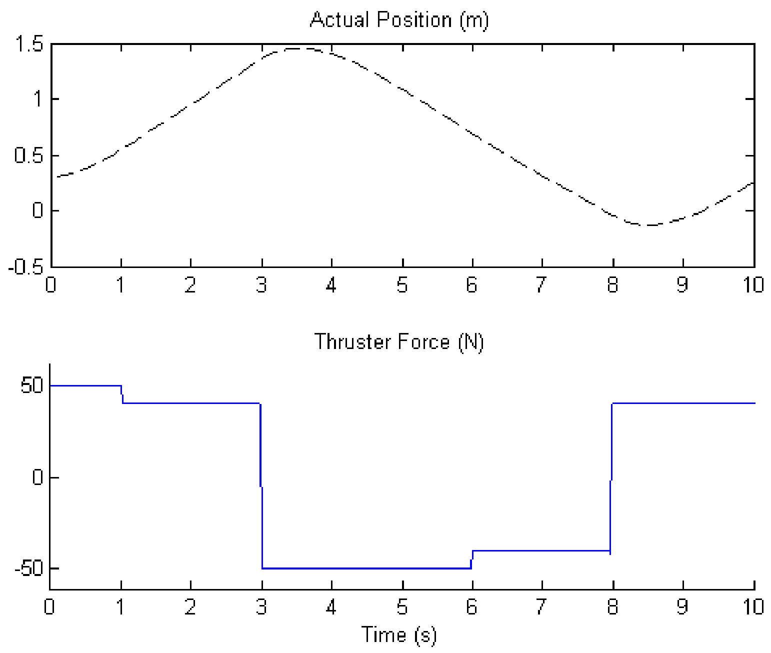


**Figure 3.9:** Response of the simulated actual vehicle in 1-DOF, given the thruster input sequence, with no time delay.

For this test, a ghost vehicle was introduced with some vehicle model parameter mismatches as well as an unmodeled ocean current of 30 N in the positive y-direction. With no prediction implemented, the uncorrected ghost response, compared to the actual vehicle, is shown in Figure 3.10. As can be seen from this figure, the ghost error grows very large since it does not consider the actual state (this would be an open-loop form of control with the operator only controlling the ghost vehicle). Although undelayed, the uncorrected ghost vehicle does not provide sufficient accuracy to control the actual vehicle.

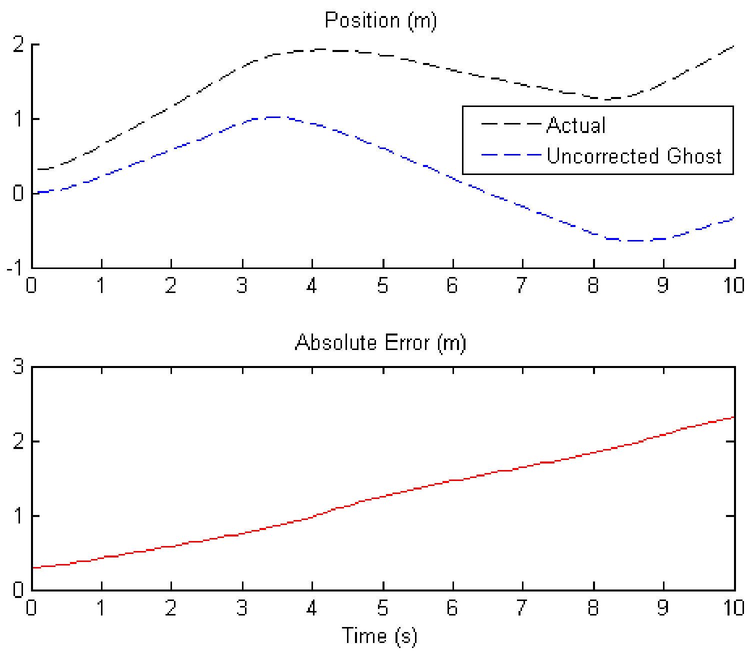


**Figure 3.10:** Uncorrected ghost vehicle response with vehicle model mismatches and an unmodeled ocean current of 30 N in the positive y-direction.

The next test implements the predictor system as discussed in §2.4.2. To ensure the predictor system works, it is first tested with no time delay, so every time a state measurement is received from the actual vehicle, it is immediately used to update the ghost vehicle (see Figure 3.11). As expected, after the first measurement is received, the ghost vehicle perfectly tracks the actual vehicle even with the model mismatches and disturbance. With no delay and the high acoustic bandwidth afforded, the predictive controller allows precise control of the vehicle (assuming accurate measurements from the remote vehicle).

Next, the time delay, $T_D$, is increased to 1 s (corresponding to a range of 750 m) to test the prediction accuracy of the predictor system (see Figure 3.12). Here, every time a state measurement arrives ($T_{\text{meas}} = 1\,\text{s}$), it is used to predict the future state 1 s ahead. This prediction is then used to correct the state of the ghost vehicle resulting in the sawtooth pattern as shown in the figure. Although not a desirable effect, the improved accuracy of the prediction is of greater importance; additionally, the corrections, or jumps, are only a

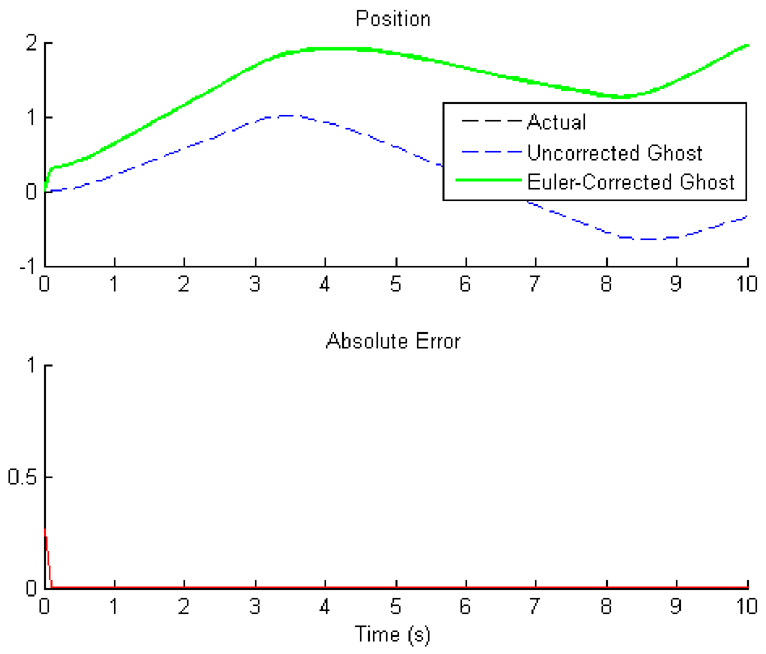


**Figure 3.11:** The predictor system implemented to update the ghost vehicle with no time delay ($T_D$ = 0 s).

short distance (below the correction threshold, $\varepsilon_c^{\max} = 0.5\,\mathrm{m}$, chosen earlier). Also note that this test presents a worst-case scenario because the unmodeled disturbance of 30 N represents a relatively strong ocean current ($\sim 0.3\,\mathrm{m/s}$). Improvements could be made to the prediction if this current was estimated using online filtering techniques such as Steinke's method and the Kalman filter [21].

Finally, the predictor is tested with a time delay of 3 s, corresponding to a vehicle range of ~2.3 km. The response of this test is shown in Figure 3.13. As is shown on the plot, the prediction error at each correction step is approximately equal to the threshold, $\varepsilon_c^{\max} = 0.5\,\mathrm{m}$, set earlier. Any increase in the time delay would cause the prediction error to be constantly over this threshold. As such, the maximum, unaided prediction time is 3 s, given the chosen parameters and model accuracy. The MATLAB code used to obtain these results is listed in Appendix A.3.

### Semiautonomous Auto-Op Mixer

The final component to be tested is the auto-op mixer and the combination of predictive and autopilot control schemes. To properly test this a simulation was built in 2-DOF. Also, for consistency, the human operator in these tests was replaced by a PID virtual operator,

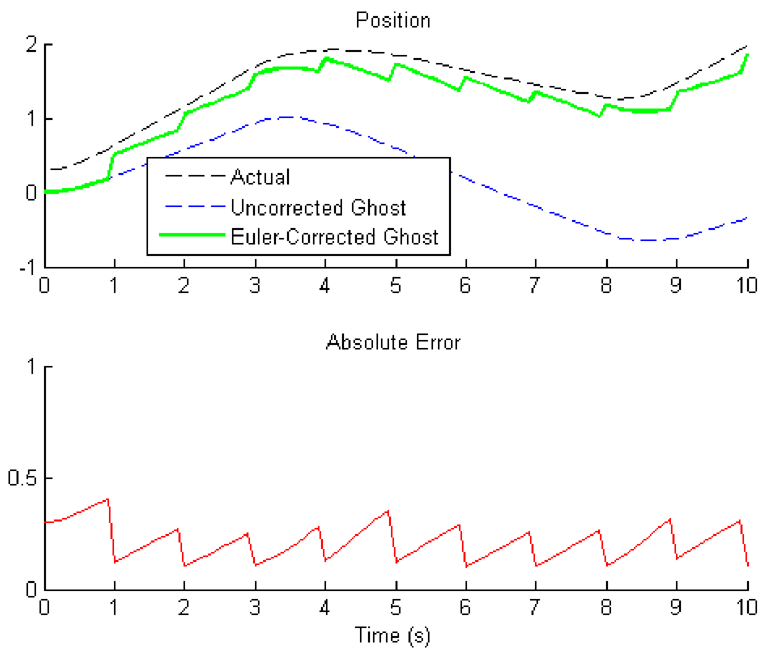


**Figure 3.12:** The predictor system implemented to update the ghost vehicle with $T_D$ = 1 s.

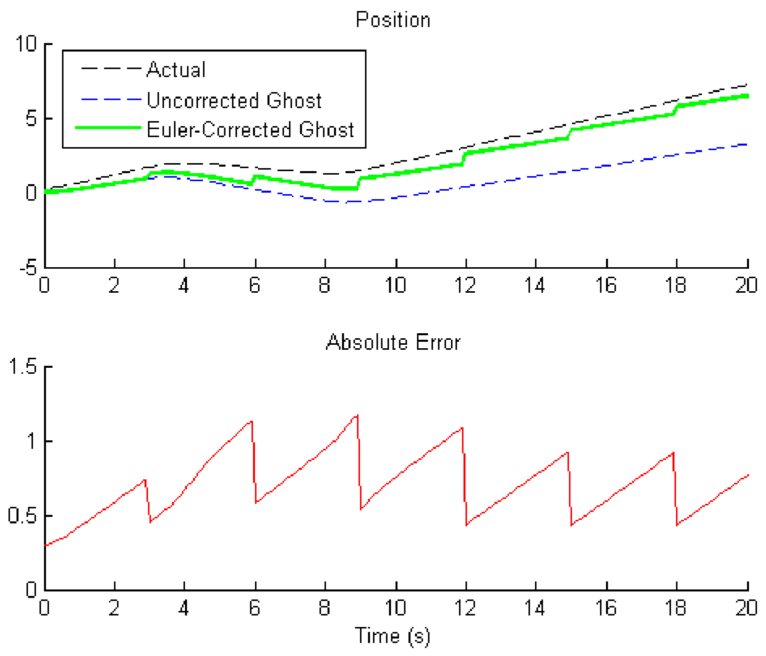


**Figure 3.13:** The predictor system implemented to update the ghost vehicle with $T_D$ = 3 s.

like in [21]. The goal of the test is to successfully pilot the vehicle to the waypoint given varying time delays and an unmodeled disturbance of, again, 30 N. The setup is the same as the PID tuning setup shown in Figure 3.3 with the added disturbance in the positive y-direction.

First, the mixer is tested with no time delay. From Figure 3.11, the prediction error with no delay is negligible so the mixer should stay in manual control only ($K_\varepsilon = 0$). This is indeed confirmed in Figure 3.14 where the ghost almost perfectly tracks the response of the actual vehicle with a small mixing factor. Here, the operator alone is responsible for piloting the vehicle and counteracting the ocean current, comparable to ROV operation. Figure 3.15 shows the case where the time delay has been increased to 1 s. Notice here that, although the predictive-only mode is controllable, the auto-op mixer still augments the virtual operator commands with autopilot commands due to slightly increased prediction error. This also reduces the effect of the ocean current on the actual vehicle since it attempts to track the ghost vehicle through waypoints.

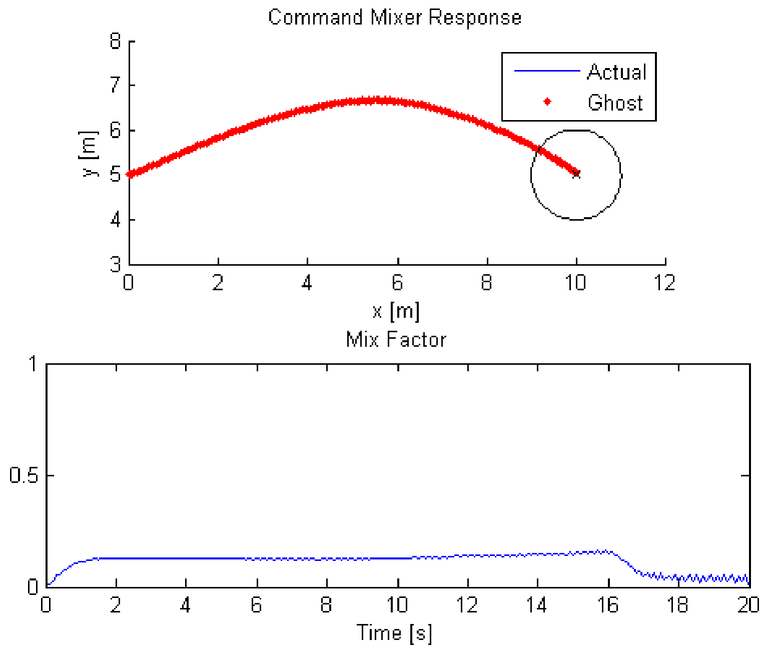


**Figure 3.14:** The mixer 2-DOF test implemented with no time delay and strong ocean current.

Next, we analyze the time where predictive-only control breaks down ($T_D = 3\,\mathrm{s}$) as shown in Figure 3.16. With the added benefit of the auto-op mixer, the virtual pilot is still able to pilot the vehicle to the goal, despite the large time delay. Around the 3 s mark in

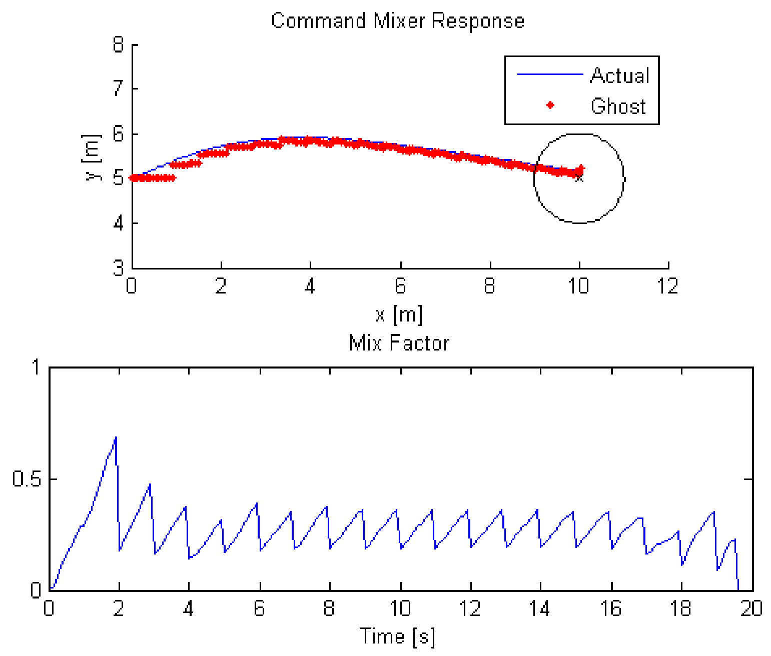


**Figure 3.15:** The mixer 2-DOF test implemented with a 1-s delay.

the figure, it is observed that the mixer is in autopilot-only mode for a short period. This is due to the large current and long prediction span creating excessive prediction error over the threshold of 0.5 m. Once the autopilot corrects this error, the mixer once again uses the virtual pilot's commands.

The main goal of the auto-op mixer is to assist the operator and augment their commands in the presence of unmodeled disturbances or model mismatches. The previous tests illustrate that the mixer works adequately with the chosen parameter values. Further research may identify an analytical or stability-based method to better determine these parameters, increasing the performance of the mixer. For this research, the manual method described is considered sufficient. With the successful tests in 1-DOF, the complete control continuum may now be implemented and tested in the full simulator environment. The code for the MATLAB simulation of the auto-op mixer is listed in Appendix A.4.

## 3.3 TUV Simulation Implementation

A full simulation environment, complete with simulated vehicles, worksite, and operator station, is required to fully test the usefulness of this controller. The simulator must meet

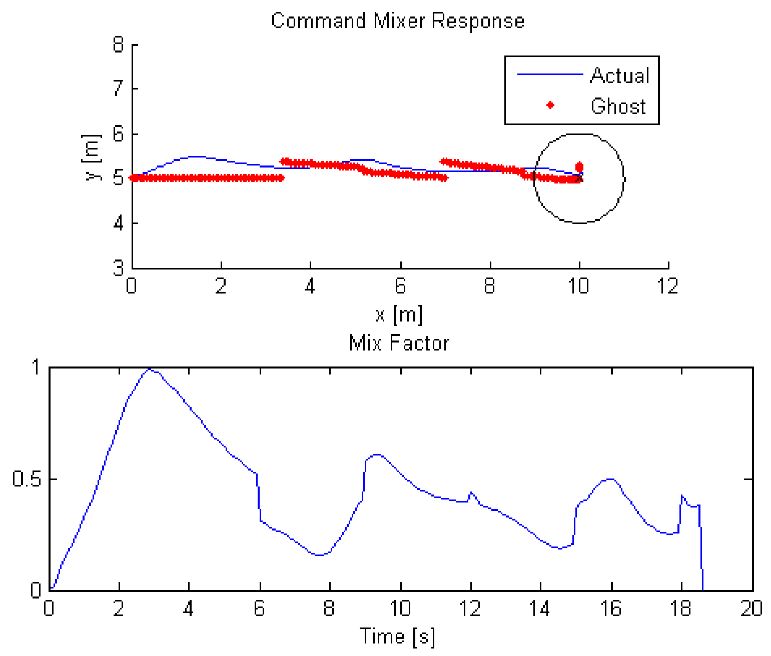


**Figure 3.16:** The mixer 2-DOF test implemented with a 3-s delay.

the following goals:

1. To simulate the remote and ghost vehicles in an underwater environment and provide the operator with visual feedback from both.
2. To present this visual feedback to the operator through an easy-to-use control interface that provides a simulated scene and controller customization.
3. To accept real-time commands from the operator in a similar manner as existing ROV controllers.
4. To introduce a simulated communications time delay between the operator and remote vehicle.
5. To implement the proposed control continuum and control the remote vehicle using the ghost vehicle.
6. To present the operator with a representative task to complete at various time delays.

To build a computer simulation, the problem must be broken into functional blocks that can communicate with the main controller. This is done by constructing a context diagram of the system, based on the goals for the simulator (see Figure 3.17). There are a number of pre-existing underwater vehicle simulators on the market such as *ROVsim Pro* [49] and *SubSim* [50]; however, many of these solutions are very costly or could not be easily adapted to accomplish all of the goals outlined. As such, it was decided to implement a completely custom simulation environment using Microsoft Robotics Developer Studio® (MRDS) [51] as the foundation. The following sections introduce the MRDS environment and how it can be used to meet the goals. Finally, the implementation of the control continuum in the simulation environment is discussed.

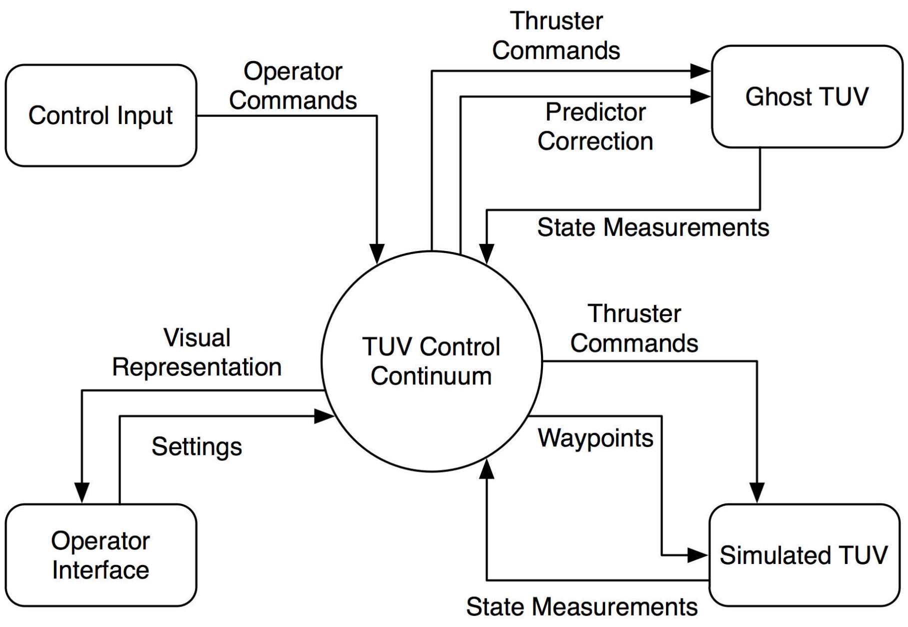


**Figure 3.17:** Simulator context diagram showing the interaction between the main controller and the other components of the system.

## *3.3.1 MRDS Overview*

MRDS is a suite of technologies combined into one development environment to create real-time simulations of robotic vehicles. It pays special attention to land-based, mobile robots, but with some modification, can also be used for underwater robotics. One of the main reasons for choosing this package is the integration of the Concurrency and Coordination Runtime (CCR) and Decentralized Software Services (DSS) libraries. Together, these libraries allow for concurrent and asynchronous processing between the many services required to simulate the robot and controller. With respect to the context diagram (Figure

3.17), each component can be realized as a standalone service using the DSS library with message passing employed for the component interaction. This feature also allows the services to be distributed among multiple computers, meaning the control system could be easily ported to an actual vehicle. The simulator also comes with a built-in visual simulation environment based on the XNA framework. This includes an extensive physics engine and tightly integrates with the other services. Lastly, the MRDS supports user input from a multitude of input methods ranging from virtual devices, to physical hardware connected to the PC. With these features, MRDS is a good candidate to fulfill all of the goals required to test the control continuum.

### *3.3.2 Building an Underwater Simulation*

The focus of this section is to discuss how MRDS can be used to properly simulate the visual effects, physics, and representations of the vehicles and the environment. As mentioned, MRDS is not expressly designed for underwater scenes, so some customizations and modifications are required to accurately model them.

#### Visualization Aids

For the simulation to be effective, an accurate 3D representation of the scene needs to be presented to the operator. In terms of visualization aids, the following are required:

1. Simple vehicle model.
2. Water effects (turbidity).
3. Terrain model.
4. Objects of interest for interaction.

A simple generic vehicle model was constructed and loaded into the MRDS environment (see Figure 3.18). It is meant to mimic a simple, open frame vehicle and to give the operator a visual guide to control the vehicle. The ghost vehicle is simply presented as a transparent version of the same model. As mentioned before, the coordinate system of the controller was chosen to match that of the simulation and is shown in Figure 2.6.

Next, since MRDS is not meant for underwater scenes, the turbidity of ocean environments was simulated using a simple fog model. This helps to limit the visibility of

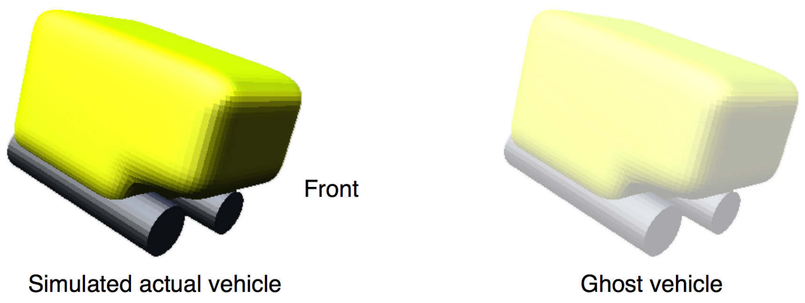


**Figure 3.18:** The generic vehicle visual model used by the simulator.

the operator, similar to an actual underwater environment. A simple terrain model was also loaded into the simulator to provide a frame of reference for control. In practical applications, the actual bathymetric maps would be loaded into the simulator. Finally, boxes are added to the scene to create a simulated obstacle course representing a narrow corridor a vehicle might be required to navigate through. The underwater scene with the water fog effect, the terrain model and the obstacle course is shown in Figure 3.19.

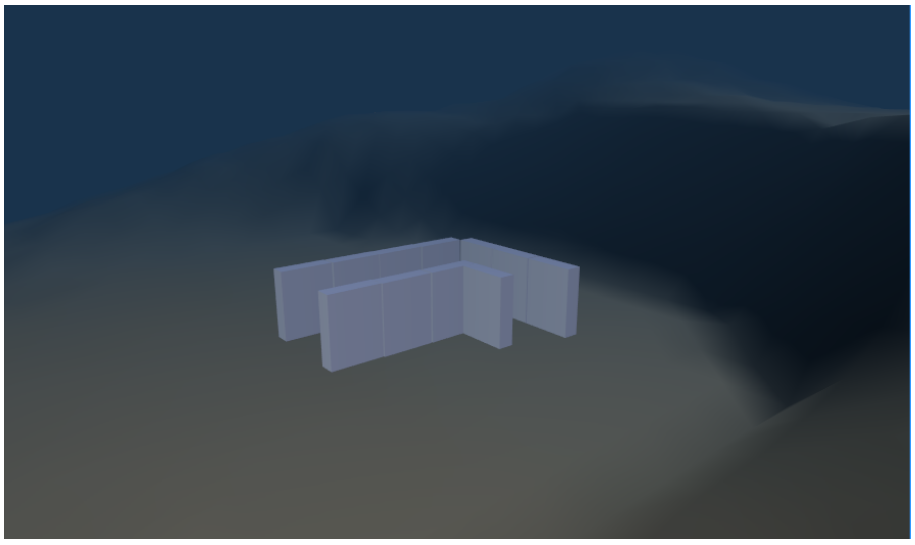


**Figure 3.19:** Simulated underwater scene with obstacle course.

### Simulating Buoyancy

Another aspect of the simulation that must be manually introduced is the simulation of buoyancy. Since MRDS is for land-based robots, it only directly supports a gravitational force. An intuitive solution would be to manually add a buoyancy force to the robot equal in magnitude to the force of gravity, but extending from the centre of buoyancy (see Figure 3.20). Adding this force would allow the built-in physics simulation to automatically determine the self-righting moments. Unfortunately, this did not work when adding it to the physics simulation due to a synchronization issue with the gravity force causing the vehicle to continuously rise. Instead, the simulation's gravity is disabled and the two forces in Figure 3.20 are manually added to the vehicles. Although partly bypassing the built-in physics simulation, it produces the desired effect and allows the vehicle to self-right.

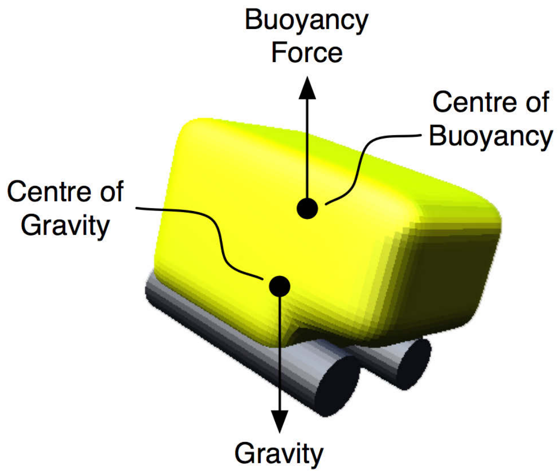


**Figure 3.20:** Ideal representation of buoyancy force.

### Vehicle Motion

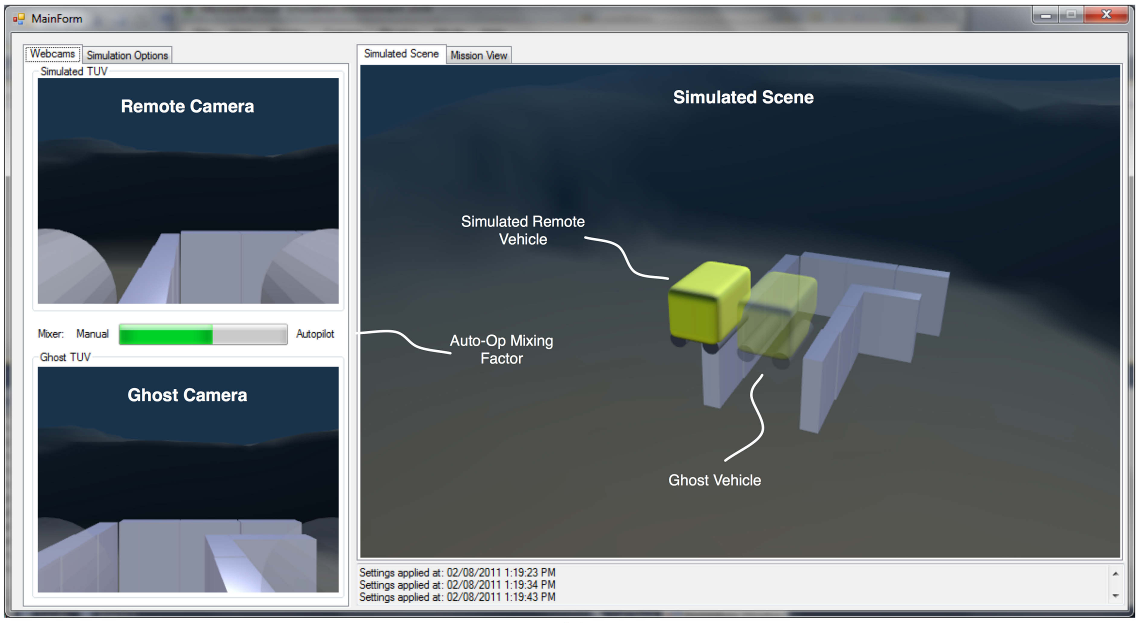
The final part of the physics simulation that needs modification is the vehicle motion model. For the simulated "actual" vehicle, the motion will be determined by the built-in physics engine with the thruster forces from the configuration in §2.3.5, and the additional drag forces from the vehicle dynamics. If an actual vehicle is available, then the simulated TUV's state will be updated immediately and as frequently as possible with state measurements from the remote vehicle. For the ghost vehicle prediction, a manual Euler method is added to the simulation. One problem is that the orientation of the vehicle, reported by the built-in physics engine, is returned as a quaternion. Quaternions are an extension of complex numbers and a common way to represent orientations/rotations in 3 DOF while avoiding problems like gimbal lock. Instead of applying the Euler prediction method to the quaternion directly, the relative Euler angles are calculated for the prediction span. The simulator has built-in methods to then convert the relative rotations to a new quaternion ($q_{\text{rot}}$). Multiplication of two quaternions is equivalent to a rotation, so to get the new orientation of the vehicle, the original quaternion ($q_0$) is right-multiplied by the relative rotation quaternion:

$$q = q_0\, q_{\text{rot}} \tag{3.1}$$

## *3.3.3 Operator Interface and Control*

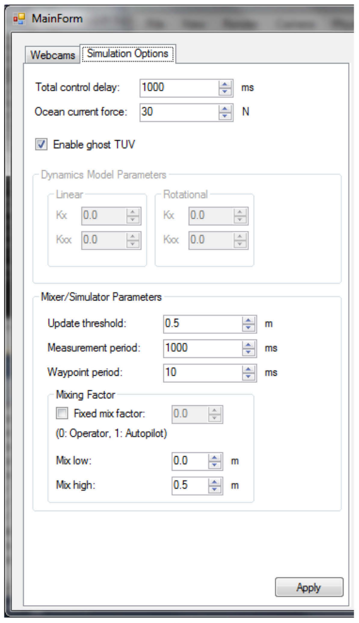
In order for the operator to interact with the controller and adequately control the remote vehicle, an effective operator interface is required. The interface should present the operator with the full simulated scene, feedback from simulated cameras on each vehicle (plus an actual camera feed if a real vehicle is available), any required information about the control system itself and the ability to adjust controller parameters as desired. The interface was constructed with the built-in MRDS utilities and is shown in Figure 3.21. Figure 3.21(a) shows the main view of the interface with the simulated 3D scene including the simulated remote and ghost vehicles, and the views from the simulated cameras and real cameras if any exist. It also displays the current auto-op mixing factor, $K_\varepsilon$, to indicate if the remote vehicle is currently placing more emphasis on operator or autopilot commands. This layout would be most often used by the human operator to complete the mission's tasks. The figure also shows a Mission View tab on the interface. This is a place holder for a waypoint editing interface that is not implemented for this thesis, much like the mission planners used for AUV systems. Figure 3.21(b) shows the simulator/controller settings panel of the interface. This can be used to manually adjust the simulation environment (such as ocean current or control time delay), or to manually tune the control continuum (such as the measurement period, $T_{\text{meas}}$, or the auto-op mixer parameters, $\varepsilon_p^{\text{low}}$ and $\varepsilon_p^{\text{high}}$).

The other important aspect of how the operator interacts with the simulator is the operator command input. MRDS has many solutions for this such as virtual control consoles and the ability to interact with connected hardware. Due to the ease of use and flexibility, an Xbox 360® gamepad is used to issue operator commands. MRDS has built-in support for this type of controller and it is similar to current methods for controlling ROVs. A summary of the controls using the gamepad is shown in Figure 3.22. Each joystick DOF of the controller is mapped to vehicle thrusters such that a force is produced in the indicated vehicle DOF. This allows the operator to control the vehicle in each actuated DOF and the joysticks allow for variable thrust power.



```
Settings applied at: 02/08/2011 1:19:23 PM
Settings applied at: 02/08/2011 1:19:34 PM
Settings applied at: 02/08/2011 1:19:43 PM
```

(a)



Total control delay: 1000 ms
Ocean current force: 30 N
[✓] Enable ghost TUV

Dynamics Model Parameters

| Linear | | Rotational | |
|---|---|---|---|
| Kx | 0.0 | Kx | 0.0 |
| Kxx | 0.0 | Kxx | 0.0 |

Mixer/Simulator Parameters
Update threshold: 0.5 m
Measurement period: 1000 ms
Waypoint period: 10 ms
Mixing Factor
[ ] Fixed mix factor: 0.0
(0: Operator, 1: Autopilot)
Mix low: 0.0 m
Mix high: 0.5 m

(b)

**Figure 3.21:** Screenshot of the operator interface used to interact with the simulated vehicles and controller: (a) view of the simulated scene and camera feeds from the simulated vehicles; (b) the settings for the simulator.

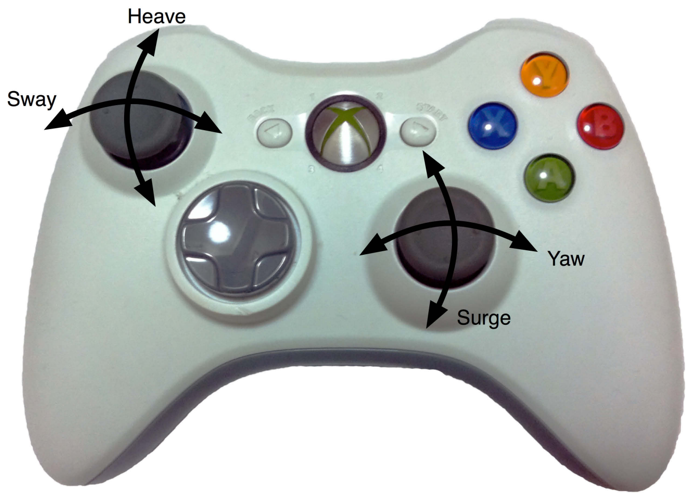


**Figure 3.22:** Gamepad motion mapping for user command inputs.

### *3.3.4 Control Continuum Implementation*

The final component to implement in the simulation environment is the control continuum itself. The key to its implementation is to determine the best software service model. MRDS services use a message-passing scheme facilitating asynchronous communications through a subscription model. That is, one service subscribes to notifications from another service. Once a notification is received, it is added to a message queue and handled concurrently with other processes in the service. Services can also manually push messages to other services as long as a corresponding port is listening for the message. With respect to the context diagram in Figure 3.17, each component including the control continuum is implemented as its own service. The continuum service subscribes to notifications from all of the external components. For example, when a state measurement is ready from either the ghost or simulated TUVs, they issue a notification with the state as the message payload. Since the continuum is subscribed to these messages, it receives the state and can process the information concurrently with other processes (such as receiving input from the operator). Once the processing is complete the continuum can selectively send messages to the desired service, such as with the prediction calculation and sending a message to the ghost vehicle service with the predicted state which is then used to update the visual representation. A simulated time delay is also implemented in the control continuum through the message passing scheme. MRDS provides a special timeout port that messages can be posted to after a specified timeout period. This timeout period is set to the round-trip time dely, $T_D$, and is implemented on the simulated vehicle (similar to Figure 2.11).

The continuum itself is implemented similar to the proof-of-concept tests in §3.2. The difference here is that all DOF are implemented and it is executed in real time with control from a human operator. With respect to the context diagram, the two main data flows are illustrated in Figure 3.23. These processes run concurrently, so the ghost vehicle is being updated at the same time the simulated TUV is calculating the auto-op mixer commands. For the full simulation, when the continuum is in the close control phase, the operator does not need the ghost vehicle and can turn it off if they wish. When the time delay exceeds ⅓ s and the continuum evolves to the predictive phase, the ghost is introduced to aid the operator (the pseudocode for the predictive phase is shown in Algorithm 3.1). For all time delays above this (for this thesis) the auto-op mixer handles the commands sent to the vehicle and the data flows in Figure 3.23 are in full effect.

**Algorithm 3.1** Prediction algorithm

```
 1: READ initial state from simulated TUV state measurement
 2: TRANSFORM all velocities to body-fixed frame
 3: Remove stale commands from queue
 4: for each command in queue do
 5:     GET command time span
 6:     for each Euler time step in command time span do
 7:         GET acceleration from operator command
 8:         CALCULATE drag and buoyancy forces
 9:         CALCULATE Euler prediction step
10:         ADD current step to overall prediction
11:     end for
12: end for
13: TRANSFORM predicted velocities to inertial frame
14: TRANSFORM relative Euler angles to quaternions
15: if correction distance $\leq \varepsilon_c^{\max}$ then
16:     SEND predicted state to ghost vehicle
17: end if
```

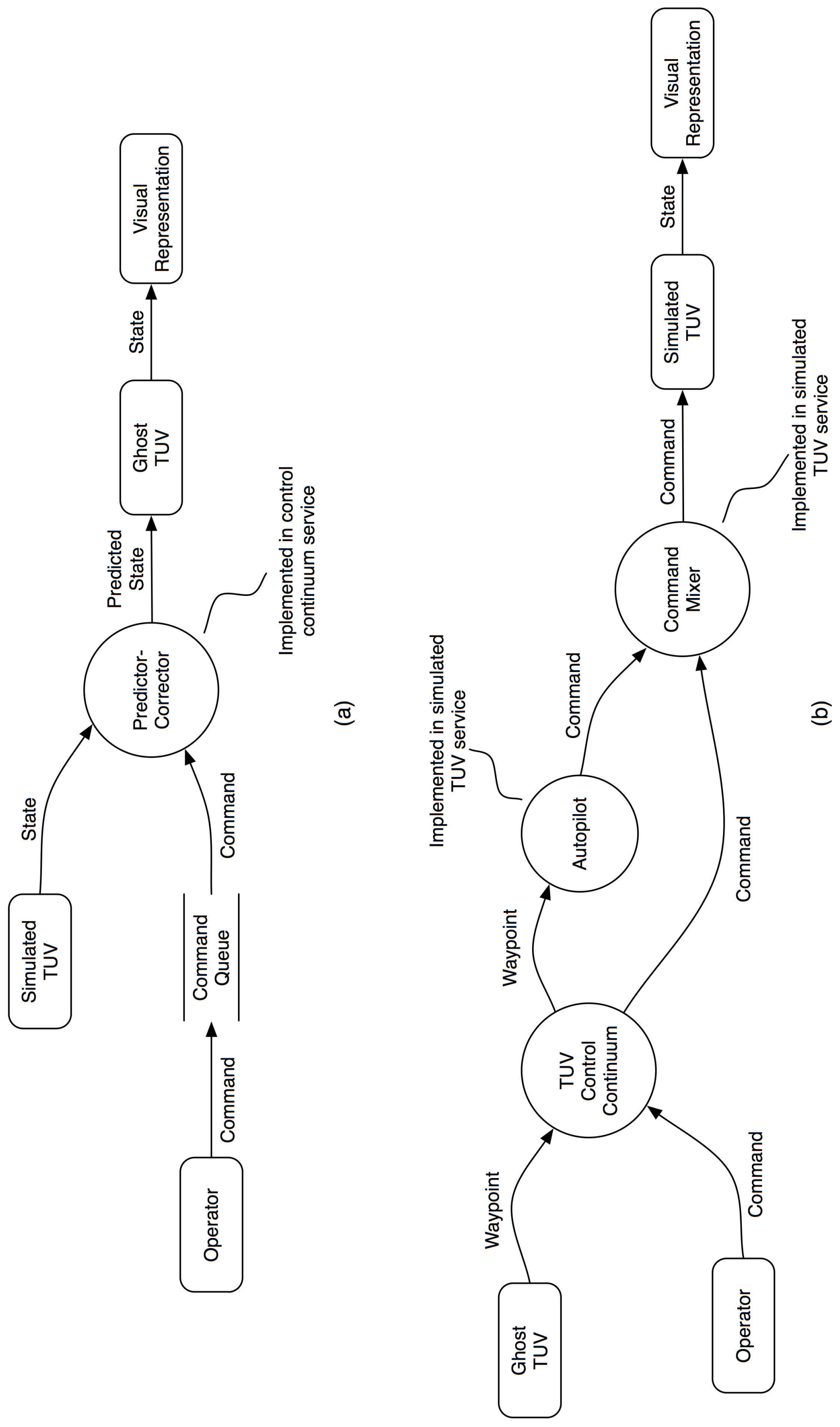


**Figure 3.23:** Data flow diagrams of the two main processes: (a) the ghost prediction and correction data flow; (b) the command mixer data flow. These processes run concurrently to one another on their corresponding services.

# CHAPTER 4

# TUV SIMULATOR RESULTS AND DISCUSSION

In the absence of an actual test vehicle (a vehicle was available, but it could not be made ready in time to test this controller), any experimental testing is limited to the simulator and the tests implemented therein. With the vehicle dynamic model introduced in §2.3 and the full simulator environment, the author feels the virtual scene presented to the operator is a good representation of reality and is a suitable test bed for the controller introduced in this thesis. With the kind support of Dr. Mae Seto from DRDC Atlantic, the simulator was tested by an actual ROV pilot, Mr. Mark Rowsome. Mr. Rowsome has considerable experience flying ROV systems and was part of the ROV team responsible for the recovery efforts of the Swissair Flight 111 incident in 1998. With this experience, Mr. Rowsome provides invaluable insight, comparisons with existing systems and can verify the usefulness of the controller through his feedback.

The goal of this section is to test the effectiveness of the control continuum as implemented in the simulator and gauge the ease in which the operator can control the vehicle. First, the operator interface is tested to ensure it adequately conveys the state of the controller and vehicles to the operator and facilitates easy control. Next, the control continuum is tested and the parameter tuning methods described in §3.2.2 and §3.2.3 are verified and the effects of improper tuning demonstrated. Finally, the complete system is tested with the virtual obstacle course introduced in §3.3.2. Although only simulated, these tests should provide a starting point for testing and verification of the control continuum.

## 4.1 Operator Interface

The operator interface (shown in Figure 3.21) must accurately convey the state of all important components to the operator. This must be done in real-time and updated at a sufficient frequency. The simulation physics update frequency must be several orders of magnitude higher than the vehicle response (much like the requirement for the Euler method). Additionally, the graphical update frequency must also be sufficiently high. This is measured in rendered frames per second, or FPS (equivalent to Hz), and also must be at least 15 FPS to appear smooth to the human observer. The MRDS environment allows the designer to specify the desired frame rate of each simulated camera; so, to help reduce the computational load for graphics and allocate more for physics, each camera was limited to 33 FPS. This, however, does not guarantee the frame rate, so each camera must still be tested. With respect to Figure 3.21, Table 4.1 summarizes the measured frame rates for each of the simulation's camera's. Similarly, Table 4.2 summarizes the measured update frequency of the physics engine for each vehicle. These are average values but they did not fluctuate under various conditions.

**Table 4.1:** Measured frame rates of each simulated camera.

| Camera | Frame Rate (FPS) |
|---|---:|
| Simulated Scene | 33 |
| Remote Camera | 32 |
| Ghost Camera | 29 |

**Table 4.2:** Measured physics engine update frequency of each simulated vehicle.

| Vehicle | Physics Update Frequency (Hz) |
|---|---:|
| Simulated "Actual" | 29 |
| Ghost | 37 |

In addition to the computational performance, the interface must properly convey all of this information. To verify this, the interface was presented to Mr. Rowsome at DRDC

Atlantic to get real-world feedback. Mr. Rowsome asserted the interface provided him with the information necessary to control the vehicle, and was similar to other ROV simulators he had experience with. Mr. Rowsome was also familiar with the gamepad input method and was quickly able to begin piloting the simulation.

The results obtained in this section confirm that the simulator can run sufficiently fast to accurately model the scene and also present the operator with up-to-date visual feedback. The physics results show that the simulation updates at intervals much smaller than the vehicle response time and the graphical results show that it can be rendered at a comparable rate. The input method is also effective and familiar for existing ROV pilots and allows the vehicle to be controlled in all actuated DOF and in real-time. Mr. Rowsome also confirmed that the ability to tune the controller on-demand is very useful for different tasks and environmental conditions. One suggestion he had to improve the tuning method was to map additional buttons on the gamepad (Figure 3.22) to disable the ghost correction or force the vehicle to either autopilot- or manual-only modes (discussed further in §4.3). With these improvements, the operator interface successfully meets goals 1–3 introduced in §3.3.

## 4.2 Control Continuum

With a functioning user interface, tests can now be performed to verify that the control continuum works as anticipated by the proof-of-concept testing in §3.2. The goal of the tests is to ensure the tuning methods in §3.2.3 hold for the simulator and that each control phase works in the full simulated environment. Again, feedback from Mr. Rowsome is used to verify the control continuum method and validate the "feel" of controlling the vehicle.

### *4.2.1 Comparison to ROV Control*

First, the simulator was set to have no time delay ($T_D = 0$) and no ocean current disturbance. The auto-op mixer was also forced to be in manual control only ($K_\varepsilon = 0$) and the ghost vehicle was disabled. This setup is meant to mimic operating existing ROV vehicles with no control aid. When Mr. Rowsome tried piloting the vehicle in this configuration, he indeed verified that the simulator performed similar to other ROV simulators he had experience with and that the control method (i.e. the gamepad) was effective. It is also important to verify the ocean current implemented; however, Mr. Rowsome believed the worst-case ocean disturbance of 30 N was too high, so the current for human pilot testing was reduced to 10 N, which Mr. Rowsome felt to be average. These results indicate that the simulator can accurately reproduce the experience for simple ROV-like control by an operator, and provides a point of reference for testing the control continuum.

### *4.2.2 Verification of Proof-of-Concept Tests*

The goal of the next series of tests is to ensure the continuum in the simulator has comparable performance to that in the proof-of-concept tests. This is done by setting up similar conditions to the tests in §3.2.3. The first test was done with no time delay ($T_D = 0$), the worst-case ocean current (30 N in the negative-X direction) and the mixer and ghost vehicle enabled. Figure 4.1 shows the results of the test with the movement of the vehicles in the XZ-plane through the obstacle course, the headings, and the prediction errors. As can be observed, the results are very similar to those presented in Figures 3.11 and 3.14. Since the delay is low the prediction error is also low resulting in the continuum staying in close control mode.

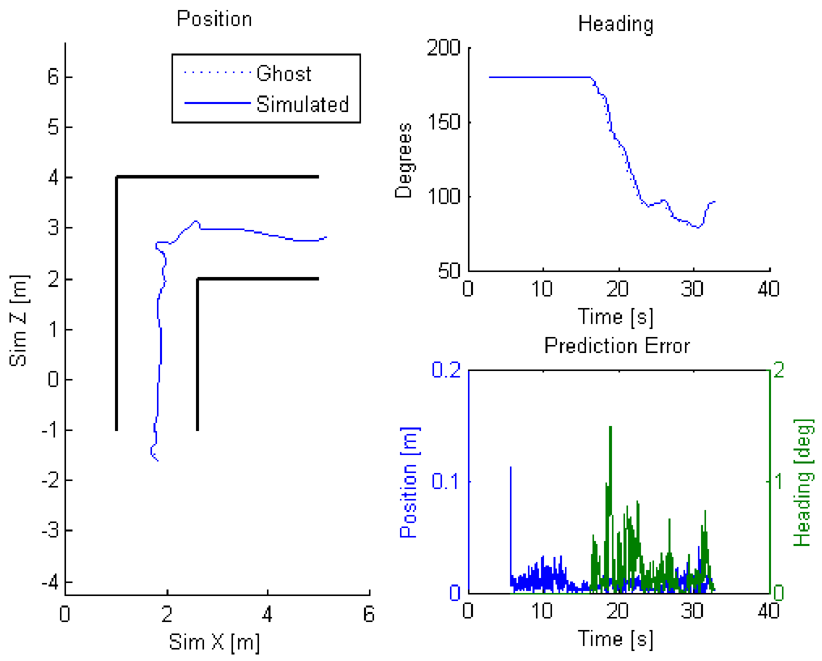


**Figure 4.1:** Control continuum test with no time delay, 30-N disturbance, and both the mixer and ghost vehicle enabled.

The results obtained from increasing the time delay to $T_D = 1\,\mathrm{s}$ are shown in Figure 4.2. Again, the results are similar to those presented in Figures 3.12 and 3.15. Here, the prediction error is increased but stays below the ghost correction threshold ($\varepsilon_c^{\max}$), so the ghost is always updated and the continuum produces a mix of operator and autopilot commands.

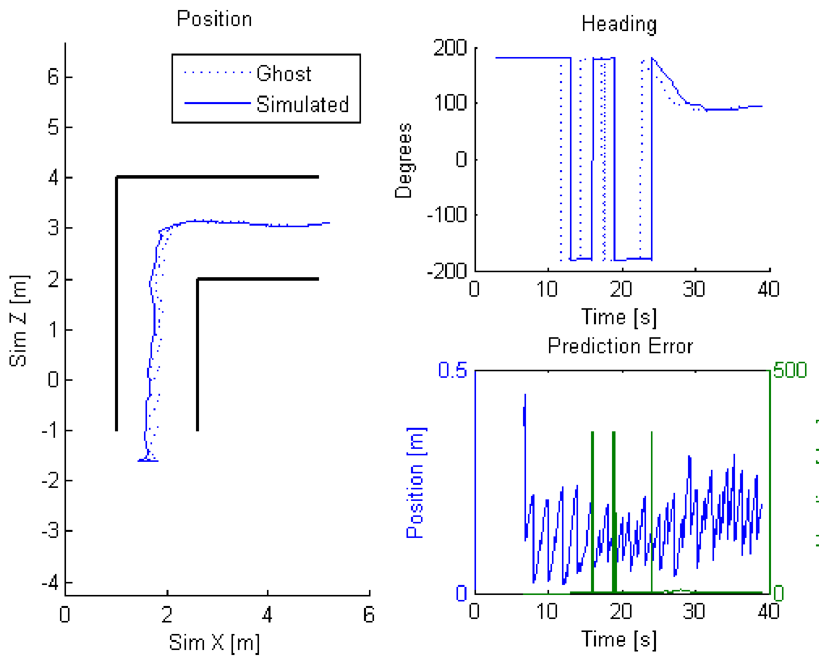


**Figure 4.2:** Control continuum test as in Figure 4.1 but with a 1-s delay.

Finally, the continuum is tested with $T_D = 3\,\mathrm{s}$ with the results as shown in Figure 4.3. Refer to Figures 3.13 and 3.16 for the proof-of-concept similarities. Here, the vehicle starts in autopilot mode since the prediction error is greater than the error threshold, $\varepsilon_p^{\text{high}}$, resulting in an auto-op mixing factor of $K_\varepsilon = 1$. Eventually, the autopilot reduces the error below $\varepsilon_p^{\text{high}}$, but at the 33-s mark, the error once again climbs past the threshold resulting in autopilot-only.

The results obtained from these tests confirm that the continuum works as expected and allows the operator to maintain control of the vehicle over varying time delays. The auto-op mixer also successfully corrects disturbance errors in all DOF and attempts to return the continuum to close control mode. This justifies the tuning methods described in §3.2.3 for the implemented simulator configuration. So, for this thesis, these tuning values will be assumed to be sufficient for all future test cases.

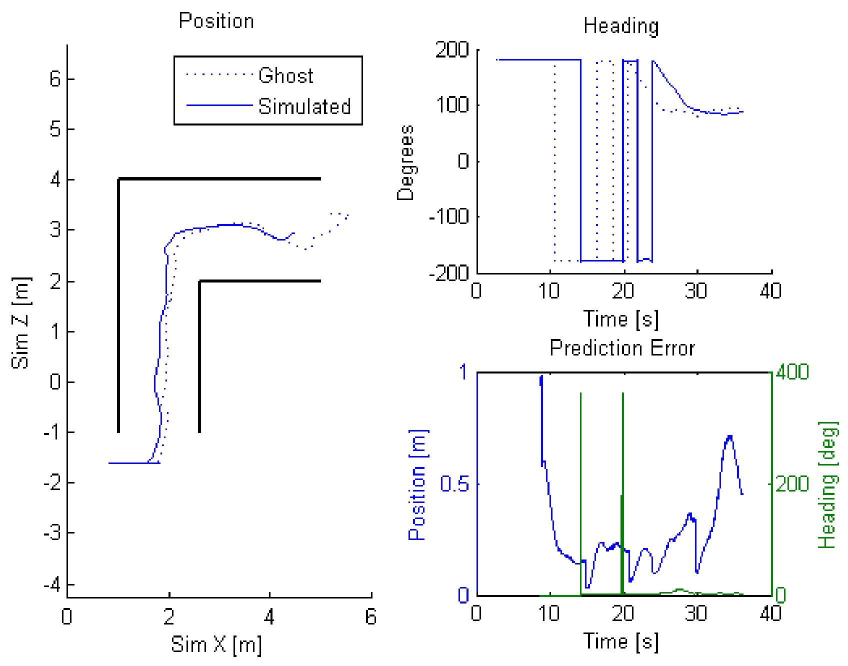


**Figure 4.3:** Control continuum test as in Figure 4.1 but with a 3-s delay.

### *4.2.3 Simulator Limits*

Even though sufficient tuning values were found, it is also instructive to analyze the limits of the controller. This section presents the effects of improper tuning and offers a sense of what the continuum can handle. First, the range limit was analyzed using the chosen tuning method. Figure 4.4 shows the response of the simulator with a 5-s time delay. The Prediction Error plot shows that the positional error is predominantly over $\varepsilon_p^{\text{high}}$. This indicates that the controller will be in autopilot-only mode most of the time for a delay of 5 s. As such, referring to Figure 2.10, $t_{s/a} = 5\,\text{s}$ (~4 km) and the continuum is in supervisory mode thereafter. At this point, the operator could choose to continue operating the ghost, or (though not implemented) could switch to the mission planning view that is familiar to AUV operators. It is also worth noting that this result is based on an unmodeled, worst-case ocean current of 30 N. If the current was estimated online using Kalman filtering techniques, such as presented in [21], then the prediction error will be reduced further (as shown in Figure 4.5).

Next, the effects of changing some of the tuning parameters were investigated. At first glance, one might consider decreasing the actual vehicle state measurement period, $T_{\text{meas}}$ (see §3.2.3), to a value below the vehicle response time in hopes to increase the accuracy

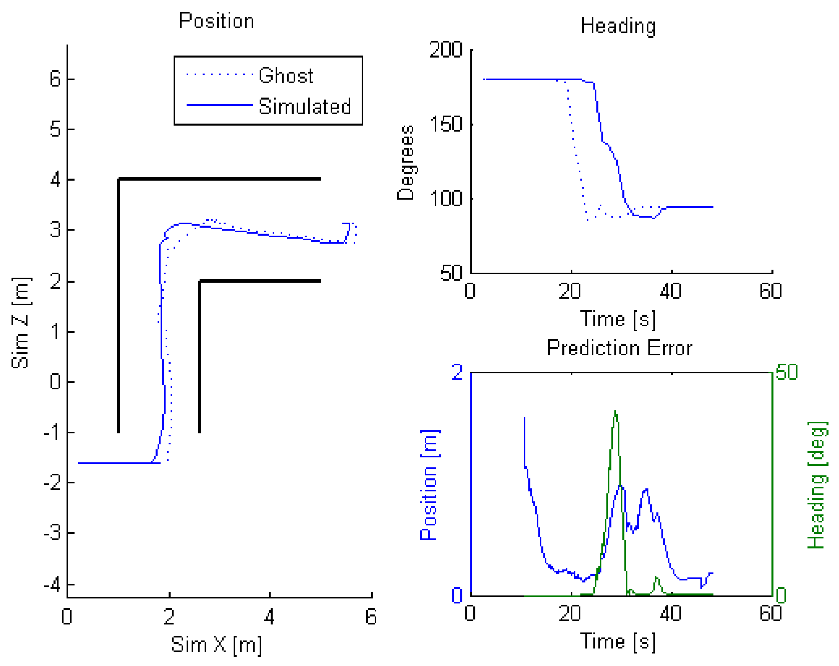


**Figure 4.4:** Control continuum test as in Figure 4.1 but with a 5-s delay to test the range limit.

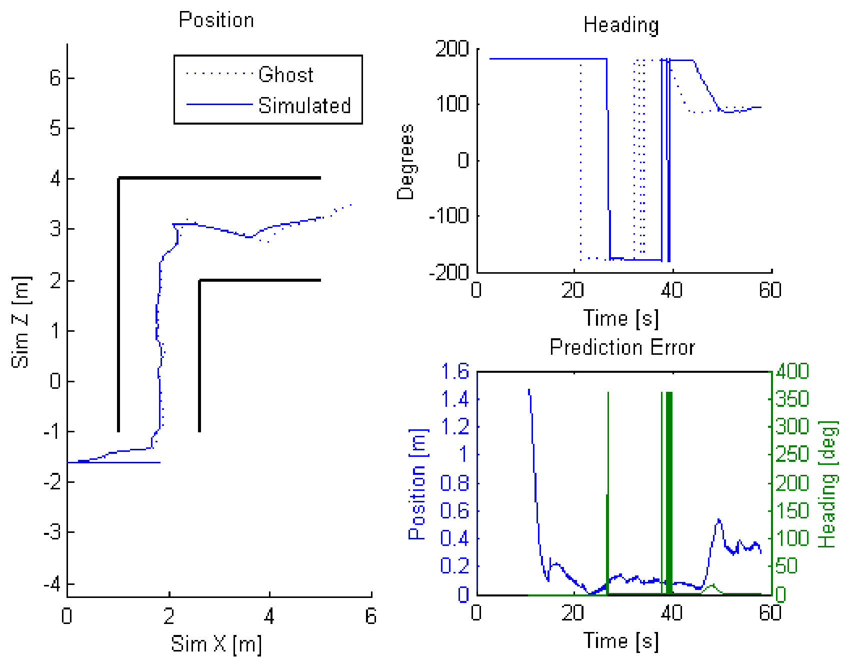


**Figure 4.5:** Control continuum test as in Figure 4.4 but with estimated ocean current in the simulator.

of the ghost correction. In theory, this would work with a perfect prediction model and sensors; however, with measurement noise and model prediction error, this results in large correction distances and makes the ghost uncontrollable. This effect is illustrated in Figure 4.6 with the ghost vehicle moving rapidly around the scene making it virtually impossible to control. A similar response arises when increasing the ghost correction threshold, $\varepsilon_c^{\max}$, as shown in Figure 4.7. Here, the ghost is corrected regardless of the correction distance. This may be useful for open-water flight, but when operating in tight quarters as in the example, these large jumps are not acceptable. The results obtained here show that the tuning parameters play a major role in determining the controllability of the ghost vehicle. The choice of parameter values can also depend on the current situation which supports the ability to manually tune the parameters using the operator interface.

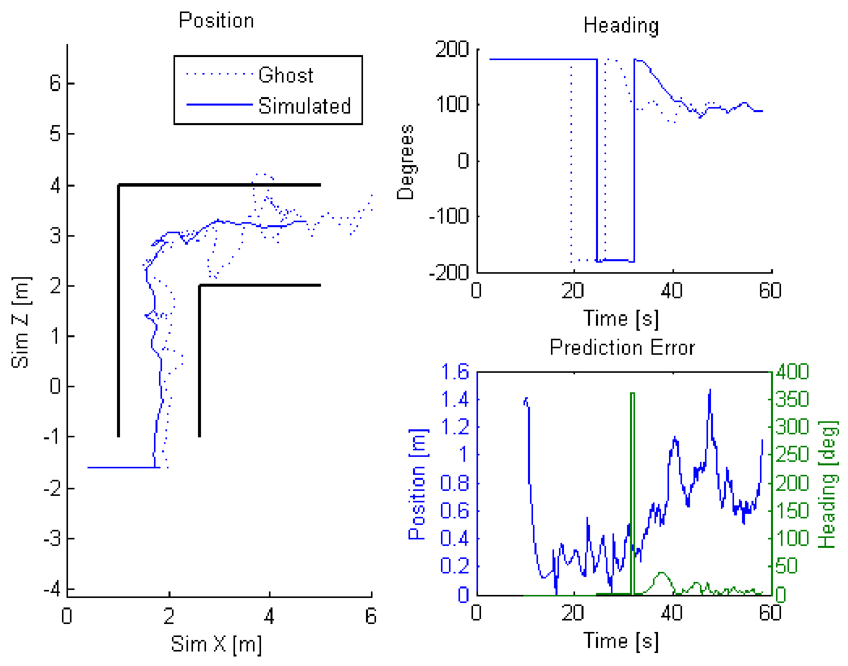


**Figure 4.6:** Simulator results with a 5-s delay, 30-N current and a ghost correction period much smaller than the vehicle response time.

## 4.3 Simulated Obstacle Course

The final test of the complete system is to analyze the performance with a small, representative task. The task is realized in the form of a small obstacle course through which

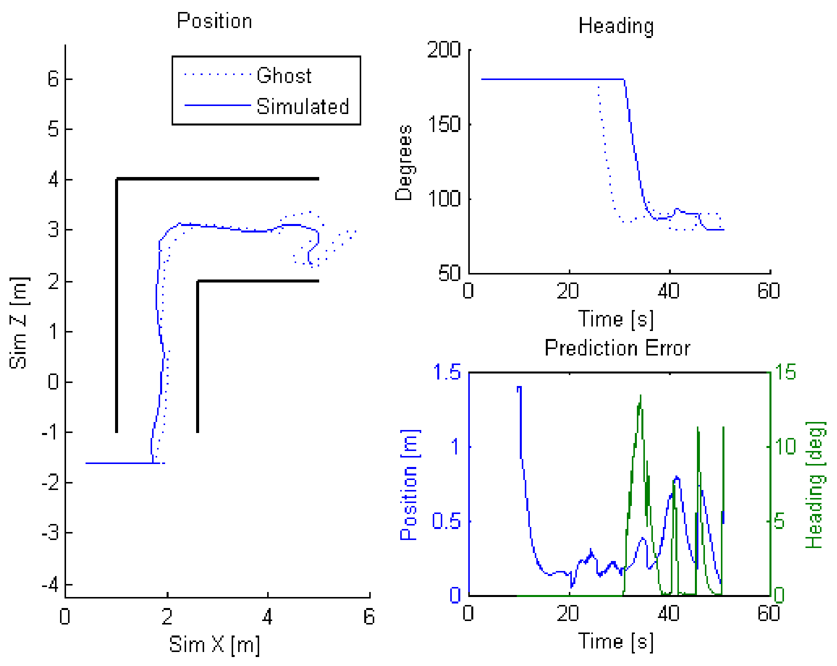


**Figure 4.7:** Simulator results with a 5-s delay, 30-N current and an unlimited ghost correction threshold.

the operator must pilot the vehicle, as introduced in §3.3.2. The course is comprised of tippable walls creating a narrow corridor with a right-hand turn. It is meant to emulate a task requiring precise motions from the operator and control in all DOF of the vehicle. The performance of the continuum with this course was measured in the time taken to fly through the course (efficiency) and the number of wall collisions (accuracy). The control continuum is considered a success if it both reduces the time taken to complete the course (with large time delays) and maintains, or reduces, the number of wall collisions. Results are presented from both the author and the actual ROV pilot, Mr. Rowsome.

The obstacle course tests were broken into three parts. First, the course was flown with no ghost vehicle or operator aid, and no ocean current. Next, the ghost remained disabled, but an ocean current of 10 N was introduced. Finally, the ghost vehicle and operator aid was enabled with the 10-N current. The current is unmodeled in the ghost vehicle so the prediction does not take the current into account. Each of these tests are flown at time delays of 0, 200, 500, and 1000 ms to gauge the effects of time delay.

For preliminary verification of the tests, the results from the author flying the course are shown in Table 4.3. The first part of the test shows the case with no ghost vehicle, and no disturbance. As predicted by theory (§2.1.1), the performance decreases with increasing time delay, breaking down at delays in the range of 500–1000 ms, as evident by the increased time and number of collisions. Part of the reason the completion time seems proportional to the time delay is because when the time delay is in the 500–1000 ms range, the operator adopted the “move-and-wait” strategy which is proportional to the delay. The next part of the test introduces the 10-N current disturbance. Similar results are observed except with increased completion time and number of collisions due to the disturbance. It should be noted that, in this case, the course was almost impossible to complete with a delay of 1000 ms (the collisions were very severe). Finally, the last part of the test introduced the ghost vehicle with the unmodeled ocean current. The preliminary results are promising since both the completion time and collisions are reduced with large time delays. In fact, the completion time is almost constant for all time delays. This makes sense since the operator is controlling the ghost vehicle which is analogous to the case with $T_D = 0$. The slight increases in times and collisions are likely due to the effects of the ghost corrections.

With successfully preliminary results, the obstacle course tests were then flown by DRDC’s ROV pilot, Mr. Mark Rowsome. The results from these tests are shown in Table 4.4. The trends observed in the preliminary results from the author are also present in Mr. Rowsome’s results. One notable difference is the similarity between Mr. Rowsome’s first and second tests, compared to the author’s who had a stronger impact from the ocean current. This is due to Mr. Rowsome’s experience in dealing with ocean currents and the fact that he adopted the “move-and-wait” strategy regardless of the time delay and was very meticulous in the way he piloted the vehicle. However, since he had no experience with time delay, he had a difficult time running the course with large delays and with the added disturbance, was unable to complete the course with $T_D = 1000\,\mathrm{ms}$. The ghost vehicle in the last test was able to alleviate some of the difficulty encountered with the large time delay. As it was his first time using this method, it took some time for Mr. Rowsome to become accustomed to controlling the ghost vehicle. In fact, he still used the “move-and-wait” strategy with the ghost vehicle with $T_D = 1000\,\mathrm{ms}$, but was at much higher frequency than the unaided cases. This resulted in improved performance with the ghost vehicle, but not as consistent as the author’s improvements since the author had more experience with the ghost control method.

**Table 4.3:** Author's results from flying the obstacle course.

| Configuration | $T_D$ (ms) | Time (s) | Collisions |
|---|---|---|---|
| No Ghost No Current | 0 | 16 | 0 |
| | 200 | 16 | 0 |
| | 500 | 22 | 1 |
| | 1000 | 37 | 4 |
| No Ghost 10-N Current | 0 | 22 | 0 |
| | 200 | 23 | 1 |
| | 500 | 28 | 3 |
| | 1000 | 54 | 8 |
| Ghost 10-N Current | 0 | 21 | 0 |
| | 200 | 22 | 1 |
| | 500 | 23 | 2 |
| | 1000 | 24 | 2 |

**Table 4.4:** Mr. Rowsome’s results from flying the obstacle course.

| Configuration | $T_D$ (ms) | Time (s) | Collisions |
|---|---|---|---|
| No Ghost No Current | 0 | 48 | 0 |
| | 200 | 50 | 0 |
| | 500 | 64 | 2 |
| | 1000 | 113 | 3 |
| No Ghost 10-N Current | 0 | 51 | 1 |
| | 200 | 52 | 0 |
| | 500 | 74 | 3 |
| | 1000 | — | — |
| Ghost 10-N Current | 0 | 52 | 0 |
| | 200 | 53 | 1 |
| | 500 | 56 | 3 |
| | 1000 | 61 | 4 |

Overall, the simulator results show promise in the control continuum method for a TUV system and fulfills the remaining goals (4–6) introduced in §3.3. The author's results verified that the continuum works and offers significant improvements over the unaided, direct-control method. Tests with Mr. Rowsome also showed that the proposed control method can be quickly and relatively easily learned by existing ROV pilots and used to complete the simulated task. From his time with the simulator, Mr. Rowsome was also able to provide some suggestions for improving the control scheme. First, he would like the ability to force the controller into a station-keeping mode. In this mode, when the operator releases the controls, the vehicle should stay stationary regardless of current. To accomplish this with the proposed controller, the mixing factor would be forced to $K_\varepsilon = 1$ and the ghost corrections disabled, $K_c^{\max} = 0$. This would cause the remote vehicle to always track the waypoints generated by the ghost and the ghost would not jump around. The addition could be easily mapped to one of the buttons on the gamepad. Conversely, Mr. Rowsome would also like to see the ability to force the ghost to predictive-only operation and allow the currents to push the remote vehicle. To implement this in the controller, the mixing factor would be set to $K_\varepsilon = 0$ and the correction threshold set to a high value (or infinity), $K_c^{\max} = \infty$. This configuration allows the remote vehicle to be pushed by the current, with the ghost vehicle being corrected regardless of the correction distance.

## 4.4 Summary of Results

The results obtained in this section mark a successful test of the simulated system. In terms of the operator interface, the results show that it adequately conveys the simulator state to the operator and facilitates real-time control of the vehicles. Testing with an experienced ROV pilot also showed that it could be easily learned with no prior knowledge of the system. The gamepad input method is both familiar and very flexible allowing for various configurations. The simulator itself also showed that it had sufficient computational and graphical performance to accurately simulate the complete system. Control continuum testing showed that the continuum performed as expected and maintains vehicle control over a wide range of time delays. Testing with two different people indicated that experienced ROV pilots can still apply some of the control methods they are accustomed to, while the author's tests showed that an inexperienced pilot can still learn the system and demonstrate similar performance results. The results also showed that care must be taken in choosing controller parameters that are appropriate for the given situation.

All combined, the system performed well when faced with the task of navigating an obstacle course. Mr. Rowsome felt the task was a good representation of a real-world scenario and a suitable test for the continuum. After becoming acquainted with the control paradigm, Mr. Rowsome felt the ghost robot and the auto-op mixer were able to improve his performance in navigating the course. This effect is evident when analyzing the results from both Mr. Rowsome and the author. In fact, Mr. Rowsome was unable to complete the course with a 1-s time delay and ocean current without the aid of the control continuum. From these tests, some conclusions can be drawn about the control continuum and operator performance:

- The continuum improves operator performance in cases of large time delay.
- The effect on the vehicle control imparted by the control continuum is analogous to maintaining close control with the actual vehicle.
- Some training may be required to utilize the ghost vehicle to its full potential (especially for existing pilots).
- The continuum successfully increases the effective close control range of the operator to $\sim$4 km (with the assumptions made).

With successful simulator tests, the controller should be easily portable to an actual vehicle as long as it meets the software requirements. No other system has been found in the literature that accomplishes a similar task for TUV systems. Assuming the assumptions in this thesis are reasonable, then the proposed control system should unlock previously unexploited potential for TUV vehicles.

# CHAPTER 5

# CONCLUSION

This thesis developed both a new control continuum for TUV systems and a simulator to test it with. The simulator allowed the control continuum performance to be tested at varying control time delays with a small representative task. This chapter first summarizes the key contributions and results found in this research and then discusses the future work required to carry the research forward.

## 5.1 Conclusions

Chapter 1 of the thesis provided a brief background of existing underwater vehicle systems and the technologies used therein. ROVs and AUVs are the two most popular type of underwater vehicle, but are meant for two different types of tasks that are not easily interchanged. TUVs show promise in consolidating the control paradigms, but have not yet been exploited to their full potential. Like AUVs, the cost reductions associated with TUVs and the removal of the tether, and possibly surface vessel, is very attractive for practical applications. This advantage is a result of the acoustic link employed by TUV systems. Current research suggests that data rate-range products can be realized in the order of 100 km*kbps with successful data transmission at ranges up to 200 km through this link. In terms of navigation, research was found suggesting vehicles should be able to position themselves with centimeter accuracy and within $5^\circ$ in orientation. Chapter 1, finally concluded that existing control and UUV systems cannot easily facilitate both close and supervisory control. The solution is to remove the physical tether and design a control system that can handle the inherent delay associate with acoustic links.

Chapter 2 developed the theory necessary to create a controller that can accomplish the research objectives. First, the problem with controlling a vehicle through time delay was outlined. It was concluded that a human operator can only maintain control at a maximum 1-s time delay and that their mental load is greatly increased. To alleviate this, the operator must be isolated from the delay through methods such as predictive displays and autonomy. The remainder of the chapter was devoted to developing a simple vehicle model and a control continuum that exploits operator isolation using both predictive and autopilot control.

In Chapter 3, a full simulator environment was developed to test the proposed control continuum. The controller required parameter tuning so this chapter outlined an *ad hoc* tuning method that produced adequate results for the given assumptions. A custom simulator needed to be developed that could meet all of the outlined requirements. At the end of the chapter, the simulator was implemented with the proposed control continuum, simulated vehicles, and a simulated underwater scene including an obstacle course.

Finally, Chapter 4 presented the results obtained through tests with the simulated environment. Mr. Mark Rowsome, an experienced ROV pilot from DRDC Atlantic, tested the simulation in addition to the author's own results. In the absence of an actual test vehicle, the simulator tests provide an adequate first-pass attempt at determining the effectiveness of the controller. The results obtained in this section can be summarized as:

1. The implemented simulator effectively conveys the state of the simulated vehicles in real-time to the operator through the operator interface. All simulated cameras can be rendered at frame rates of approximately 30 FPS, presenting a smooth visualization to the operator. The physics engine also runs at comparable speeds creating a realistic simulation of the motion of the vehicles. Since the dynamics of the vehicle are linearized, it is important for the simulation time step to be much smaller than the response time of the vehicle. This condition is fulfilled for all considered test cases. As confirmed by Mr. Rowsome, the gamepad input method is also familiar to existing pilots, and an effective way to pilot the vehicles.

2. The control continuum testing verified that the simulator is a good substitute for an actual vehicle due to its similarities to actual ROVs. This was confirmed by Mr. Rowsome by setting the time delay, $T_D$, to zero with all operator aid disabled.

The next series of tests demonstrated that the control continuum can maintain close control of the remote vehicle up to delays of 5 s, or ∼4 km range, with an unmodeled ocean current of 30 N. Beyond this, the continuum is limited to supervisory-only control (either control of the ghost vehicle or manual waypoint editing) where the range is then limited by the implemented acoustic link. The continuum results also showed that the tuning values play a major role in the stability and effective range of the controller. The method described in §3.2.3, however, yielded acceptable results over a wide operating range.

3. Testing done with the simulated obstacle course also showed promising results for the control continuum. The performance for the tests was gauged using the time taken to complete the course and the number of wall collisions. Data was gathered from both the author, and the ROV pilot, Mr. Rowsome. Both data sets demonstrated that the control continuum improved operator efficiency when faced with large time delays and ocean current disturbances. Mr. Rowsome's results showed that experienced ROV pilots can quickly learn the new control paradigm with little training and compliment their existing control techniques. On average, the control continuum provided a 50% improvement in performance at $T_D = 1\,\mathrm{s}$. Ideally, the time taken to complete the course remains constant with the control continuum aid since the ghost vehicle effectively removes the delay. This was demonstrated with the author's results, but the effect was not as obvious with those of Mr. Rowsome.

With respect to the objectives outlined in §1.2.1, the research successfully fulfilled 1–5. A control continuum was successfully designed and implemented that facilitates the smooth evolution through existing control paradigms, based on control time delay, using the novel auto-op mixer. As the time delay increases, the controller gradually imposes a higher importance on the autopilot's thruster commands, rather than those of the operator. The operator is also presented with a predictive display that increases the range of close control by effectively isolating the operator from the time delay. This controller should unlock a previously unexploited potential in TUV vehicles. It allows them to remove the burden of an ROV's physical tether, while operating at ranges comparable to AUVs, all with minimal manual reconfiguration.

## 5.2 Future Work

There are a number of avenues that warrant future work in order to carry this research forward:

1. It is paramount that tests be performed on an actual vehicle. A vehicle was donated by Dr. Mae Seto and a team at Dalhousie University, but could not be made ready in time for this thesis. It is a small, tethered vehicle, but would serve as a good test for the predictor system. It has a different thruster configuration than that outlined in §2.3.5, but the simulation could be easily adapted to match the vehicle. The vehicle also needs to be outfitted with some form of positioning system. For testing purposes, this can be a simple, visual triangulation or simple acoustic positioning system that allows the position to be reported in real-time to a PC. Instead of implementing the controller on the vehicle (as it ultimately would be in practical applications), the software could be implemented on an external PC. Since it is tethered, an artificial control delay would also need to be introduced, much like the simulator here. The goal of the test would be to replace the simulated vehicle measurements with measurements from an actual vehicle and measure the performance of the predictor and autopilot systems. With successful small-scale tests, the system could be implemented on an actual TUV vehicle and progressed forward from there.

2. A formal tuning method and stability analysis for the mixer must be found. As discussed in §3.2.3, this thesis introduced an *ad hoc* tuning method that produces adequate results over a wide range of delays. An analytical approach should be developed from the complete feedback loop to better choose the values. This is especially true for the choice of the ghost correction period, $T_{\text{meas}}$. Currently, it is set equal to the time delay, $T_{\text{meas}} = T_D$, to avoid updating the ghost too frequently when the prediction error is large. A better relationship between $T_{\text{meas}}$ and $T_D$ needs to be developed. This is also true for the ghost correction distance threshold, $\varepsilon_c^{\text{max}}$. Currently, this value is set to the operator's precision requirement for the given task. A stability analysis may yield better choices for this parameter. With the choice of $\varepsilon_c^{\text{max}}$, the mixer parameters, $\varepsilon_p^{\text{low}}$ and $\varepsilon_p^{\text{high}}$, must also be chosen to ensure the autopilot is effective.

3. The improvements suggested by Mr. Rowsome should be implemented to improve

the operator experience. The ability to toggle a station-keeping mode forces the remote vehicle to follow the ghost vehicle, provided the autopilot guidance is accurate. The forced manual mode would also enable the operator to see the effects of any environmental disturbances and allow them to manually compensate. In addition to the improvements mentioned in §4.3, Mr. Rowsome would also like to see the ability to control the camera orientation and also limit, or attenuate, the maximum thruster force (for precise control).

# APPENDIX A

# MATLAB SOURCE CODE

This appendix lists all of the MATLAB source code necessary to build the control continuum proof-of-concept testing. It includes the predictive display and auto-op command mixer. All files should be placed in the same directory including the support functions. The full simulation environment described in §3.3 is also available. Those interested may contact the author at `graham.leblanc@dal.ca` for requests and instructions on how to setup and use it. The custom simulation implementation is property of this author and may only be used for academic purposes.

## A.1 Vehicle Response

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:     tuv_ode_in.m
% Author:   Graham LeBlanc
%           Dalhousie University
% Email:    graham.leblanc@dal.ca
% Date:     9 Aug., 2011
% Description:
%   Tries to measure repsonse to step input
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all; clc;
```

```
% Time span
dt = 0.01; % Sim update period
t = 0:dt:10;

% Constants
m = 85;
k1 = 40;
k2 = 200;

% Initial conditions
x_actual = [0 0];

% External force
F_T = @(tf) 50; % Thruster

% Solve ODE with Runge-Kutta method
[T_actual,Y_actual] = ode45(@(ts,y) ...
   tuv_ode_in(ts,y,m,k1,k2,F_T),t,x_actual);

% Euler
[T,Y] = tuv_euler(@(ts,y) ...
   tuv_ode_in(ts,y,m,k1,k2,F_T),t,x_actual);

% Error
Y2 = interp1(T_actual,Y_actual,T);
error = abs(Y2 - Y);

hold on;
plot(T_actual,Y_actual(:,2),'-b');
plot(T,Y(:,2),'-g');
plot(T,error(:,2),'-r');
%plotyy(T,Y(:,2),T,error(:,2));
title('Vehicle Step Response');
legend('RK4','Euler','Error');
xlabel('Time [s]'); ylabel('Speed [m/s]')
```

## A.2 PID Autopilot With Anti-Windup Compensation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:     pid_sat_windup_thesis.m
% Author:   Graham LeBlanc
%           Dalhousie University
% Email:    graham.leblanc@dal.ca
% Date:     9 Aug., 2011
% Description:
%   This file corrects only the heading first, then does normal
%       correction
%   Also deals with saturation with wind-up compensator
%   Now saves desired heading for each WP.
%   Trying to use anti-windup as in Astrom.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all; clc;

dt = 0.1; % Needs to be less than vehicle response time
t = 0:dt:60;
% t = 0:dt:50;

% Model params
m = 85;
I = 10; % Moment of inertia
% k1 = 30.5;
% k2 = 45.5;
k1 = 40;
k2 = 200;
T_max = 50; % Max thruster force
f_c = [5;5]; % Ocean current force
f_c = [0;0];

% PID gains
K_psi_P = 0; % Set to 300 for heading correction
K_psi_D = 0; % Set to 200 for heading correction
K_x_P = 300;           % Manual tuning
K_x_I = 100;
```

```
K_x_D = 200;
K_t = sqrt(K_x_I/K_x_D)/K_x_I;  % Tracking (compensator, see Feb. ...
    21 notes)
% K_t = 0;
K = [-K_psi_P -K_psi_D 0 0 0 K_x_P K_x_I K_x_D; ...
    K_psi_P K_psi_D 0 0 0 K_x_P K_x_I K_x_D; ...
    0 0 K_x_P K_x_I K_x_D 0 0 0; ...
    0 0 -K_x_P -K_x_I -K_x_D 0 0 0];
K_rot = [-K_psi_P -K_psi_D 0 0 0 0 0 0; ...
    K_psi_P K_psi_D 0 0 0 0 0 0; ...
    0 0 0 0 0 0 0 0; ...
    0 0 0 0 0 0 0 0];

% Sequence of waypoints [x,y,theta]
P = [[0;0;0] [10;10;pi/2] [10;0;pi] [5;5;pi/2] [0;10;-pi/2]];
% P = [5;10;0];
p_i = 1;
p = P(:,p_i); % Current waypoint
r_p = 1; % Waypoint radius
t_th = 0.1; % Heading threshold
F_th = [20;20]; % Force/torque input/output error threshold

% Thruster forces
T = [0;0;0;0];

% Rotation: local -> world
R = @(t) [cos(t) -sin(t); sin(t) cos(t)];

% Force and torque as func of thrusters (local coords)
F = @(T) [0 0 1 -1; 1 1 0 0]*T;
tau = @(T) 0.5*[-1 1 0 0]*T;

% Ocean current force (World -> local)
F_C = @(th) R(th)'*f_c;

% World coords
x = [5 0]; % [pos, spd]
y = [0 0];
theta = [0 0]; % Heading
```

```
% PID error vector [x;y;theta]
f_error = @(R,x,y,dth,px,py) [R' [0;0];[0 0 1]]*[1 0 -x;0 1 -y; ...
    0 0 dth]*[px;py;1];
previous_error = f_error(R(theta(1)),x(1),y(1), ...
    p(3)-theta(1),p(1),p(2));
e_I = 0; % Integral error
e_out = 0; % commanded-actual output error

% State vectors (local) [pos, spd]
x_s = [0 0];
y_s = [0 0];

% Graphing
g_X = zeros(length(t),2);
g_X(1,:) = [x(1) y(1)];
g_V = zeros(length(t),2);
g_V(1,:) = [x(2) y(2)];
g_error = zeros(length(t),1);
g_error(1) = norm(previous_error);
g_theta = zeros(length(t),1);
g_theta(1) = theta(1)+pi/2;
g_T = zeros(length(t),length(T));
g_T(1,:) = T';

%subplot(2,1,2);
%hold on;

for i = 2:length(t)

    tspan = [t(i-1) t(i)];

    % Update thrusters via PID
    error = f_error(R(theta(1)),x(1),y(1), ...
        p(3)-theta(1),p(1),p(2));
    e_I = e_I + (error + K_t*e_out)*dt; % Windup compensator
    e_D = (error - previous_error)/dt;
    previous_error = error;
```

```
% If heading is out of threshold, correct first
if (abs(error(3)) > t_th)
    KK = K_rot;
    e_I = zeros(length(error(:,1)),1);
else % Otherwise, move normally
    KK = K;
end

KK = K; % Force K
T = KK*[error(3);e_D(3); ...
    error(1);e_I(1);e_D(1); ...
    error(2);e_I(2);e_D(2);];

% If waypoint reached, move to next
if (norm([error(1),error(2)]) ≤ r_p)
    % If last one
    if (p_i == length(P(1,:)))
        T = K*[error(3);e_D(3); ...
            error(1);e_I(1);e_D(1); ...
            error(2);e_I(2);e_D(2);];
    else % If more
        p_i = p_i + 1;
        p = P(:,p_i);

        e_I = 0; % Clear integral

        prevous_error = f_error(R(theta(1)),x(1),y(1), ...
            p(3)-theta(1),p(1),p(2));
    end
end

% Clamp thruster values
T_C = clamp_thrusters(T,T_max);

% Get new output error
e_out = [F(T_C);tau(T_C)] - [F(T);tau(T)];

% Local thruster force--assume const for update period
F_T = F(T_C) + F_C(theta(1)); % Thrusters + ocean current
```

```
    f_t_x = @(t) F_T(1);
    f_t_y = @(t) F_T(2);
    t_t = @(t) tau(T_C);

    % Get local state
    v = R(theta(1))'*[x(2);y(2)]; % Vel: world -> local
    x_s = [0 v(1)];
    y_s = [0 v(2)];

    % Euler method
    [Tx,X] = tuv_euler(@(ts,xx) ...
        tuv_ode_in(ts,xx,m,k1,k2,f_t_x),tspan,x_s);
    [Ty,Y] = tuv_euler(@(ts,yy) ...
        tuv_ode_in(ts,yy,m,k1,k2,f_t_y),tspan,y_s);
    [Tw,W] = tuv_euler(@(ts,ww) ...
        tuv_ode_in(ts,ww,m,k1,k2,t_t),tspan,theta);

    % Update state (local -> world)
    x_u = [x(1);y(1)] + R(theta(1))*[X(end,1);Y(end,1)]; % Pos
    v_u = R(theta(1))*[X(end,2);Y(end,2)]; % Vel

    % State vectors
    x = [x_u(1) v_u(1)];
    y = [x_u(2) v_u(2)];
    theta = W(end,:);

    % Save graph
    g_X(i,:) = [x(1) y(1)];
    g_V(i,:) = [x(2) y(2)];
    g_error(i) = norm([error(1) error(2)]); % No heading error
%     g_error(i) = norm(error); % Heading as well
    g_theta(i) = theta(1)+pi/2;
    g_T(i,:) = T_C';

end

% Main plot
subplot(2,2,[1 3]); hold on;
axis equal;
```

```
% xlim([-2 12]);
% ylim([-1 12]);

% Plot waypoints
plot_waypoints(P,r_p);

% Heading
% quiver(g_X(:,1),g_X(:,2),cos(g_theta),sin(g_theta),'g');

% Position
plot(g_X(:,1),g_X(:,2),'.');
title('Position');
xlabel('X [m]'); ylabel('Y [m]');

% Thruster force
subplot(2,2,2);
plot(t,g_T);
title('Thruster Force');
xlabel('Time [s]'); ylabel('Force [N]');
legend('T1','T2','T3','T4');

% Error
subplot(2,2,4);
plot(t,g_error,'r');
title('Position Error');
xlabel('Time [s]'); ylabel('Error [m]');
```

## A.3 Predictor Proof-of-Concept Testing

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:      ode_predictor_thesis.m
% Author:    Graham LeBlanc
%            Dalhousie University
% Email:     graham.leblanc@dal.ca
% Date:      9 Aug., 2011
% Description:
```

```
%   Used for predictor-corrector PoF testing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all; clc;

% Time span
dt = 0.1; % Sim update period
t = 0:dt:10;

% Constants
m = 83;
k1 = 30.5;
k2 = 195.5;
m_g = 85;
k1_g = 40;
k2_g = 200;
T_D = 0.8; % Control time delay [s]
T_M = 0.05; % Measurement period [s] (≥ dt)

% Initial conditions
x_actual = [0.3 0];
x_ghost = [0 0];
x_ghost2 = x_ghost;

% External force

F_T = @(tf) (tf≤1)*50 + (tf>1 && tf≤3)*40 - (tf>3 && tf≤6)*50 ...
      - (tf>6 && tf≤8)*40 + (tf>8)*40;
F_B = @(tf) F_T(tf) + 30; % Thruster + ocean current

% Solve ODE with Runge-Kutta method
[T_actual,Y_actual] = ode45(@(ts,y) ...
   tuv_ode_in(ts,y,m,k1,k2,F_B),t,x_actual);
[T_ghost,Y_ghost] = ode45(@(ts,y) ...
   tuv_ode_in(ts,y,m_g,k1_g,k2_g,F_T),t,x_ghost);

% Update from actual
i_update = round(T_D/dt)+1;
```

```
% Graph variables
g_x_ghost = zeros(1,length(t));
g_x_ghost(1) = x_ghost(1);
g_x_ghost2 = zeros(1,length(t));
g_x_ghost2(1) = x_ghost2(1);

%subplot(2,1,1); hold on;

% Linear extrapolation method
for i = 2:length(t)

   % Check if there's an update from actual
   if (i == i_update)
      % Get time span and update state
      tspan = [t(i-T_D/dt) t(i)];
      update = Y_actual(i-T_D/dt,:);

      % Predict the current position given delayed update
      % Still uses the erroneous model params
      [T_update,Y_update] = ode45(@(ts,y) ...
         tuv_ode_in(ts,y,m_g,k1_g,k2_g,F_T),tspan,update);
         %tuv_ode_in(ts,y,m,k1,k2,F_B),tspan,update);
      [T_update2,Y_update2] = tuv_euler(@(ts,y) ...
         tuv_ode_in(ts,y,m_g,k1_g,k2_g,F_T),tspan,update);

      % Plot the prediction
      %plot(T_update2,Y_update2(:,1),'-r');

      % Save the updated state and set new update time
      x_ghost = Y_update(end,:);
      x_ghost2 = Y_update2(end,:);
      i_update = i_update + round(T_M/dt);
   else
      tspan = [t(i-1) t(i)];

      % RK4 method
      [T,Y] = ode45(@(ts,y) ...
         tuv_ode_in(ts,y,m_g,k1_g,k2_g,F_T),tspan,x_ghost);
      % Euler method
      [T2,Y2] = tuv_euler(@(ts,y) ...
         tuv_ode_in(ts,y,m_g,k1_g,k2_g,F_T),tspan,x_ghost2);

      % Save state
      x_ghost = Y(end,:);
      x_ghost2 = Y2(end,:);
   end

   % Update plot variable
   g_x_ghost(i) = x_ghost(1);
   g_x_ghost2(i) = x_ghost2(1);
end

switch 5
   case 1
      % Actual + input
      subplot(2,1,1);
      plot(t,Y_actual(:,1),'--k');
      title('Actual Position (m)');
      subplot(2,1,2);
      hold on;
      ezplot(F_T,t);
      title('Thruster Force (N)');
      xlabel('Time (s)');
   case 2
      % Actual + uncorrected ghost + error
      subplot(2,1,1); hold on;
      plot(t,Y_actual(:,1),'--k');
      plot(t,Y_ghost(:,1),'--b');
      title('Position (m)');
      legend('Actual','Uncorrected Ghost');
      subplot(2,1,2);
      plot(t,abs(Y_actual(:,1)-Y_ghost(:,1)),'-r');
      title('Absolute Error (m)');
      xlabel('Time (s)');
   case 3
      % Actual + euler + rk4 + error
      subplot(2,1,1); hold on;
```

```
        plot(t,Y_actual(:,1),'--k');
        plot(t,Y_ghost(:,1),'--b');
        plot(t,g_x_ghost,'-b','LineWidth',2);
        plot(t,g_x_ghost2,'-g','LineWidth',2);
        title('Position');
        legend('Actual','Uncorrected Ghost','RK4-Corrected Ghost', ...
           'Euler-Corrected Ghost');
        subplot(2,1,2); hold on;
        plot(t,abs(Y_actual(:,1)'-g_x_ghost),'-b');
        plot(t,abs(Y_actual(:,1)'-g_x_ghost2),'-g');
        title('Absolute Error');
        xlabel('Time (s)');
    case 4
        % Update frequency + time delay
        subplot(2,1,1);
        plot(t,100./t);
        title('State Measurements Per Second');
        subplot(2,1,2);
        plot(t,t*2/1.5);
        title('Round Trip Time Delay (s)');
        xlabel('Range (km)');
    case 5
        % Actual + euler + error
        subplot(2,1,1); hold on;
        plot(t,Y_actual(:,1),'--k');
        plot(t,Y_ghost(:,1),'--b');
        plot(t,g_x_ghost2,'-g','LineWidth',2);
        title('Position');
        legend('Actual','Uncorrected Ghost','Euler-Corrected Ghost');
        subplot(2,1,2); hold on;
        plot(t,abs(Y_actual(:,1)'-g_x_ghost2),'-r');
        title('Absolute Error');
        xlabel('Time (s)'); %ylim([0 1]);
    case 6
        % Actual + euler + error
        subplot(2,1,1); hold on;
        plot(t,Y_actual(:,1),'--k');
        plot(t,Y_ghost(:,1),':k');
        plot(t,g_x_ghost2,'-k','LineWidth',2);
```

```
      title('Position (m)');
      legend('Actual','Uncorrected Ghost','Euler-Corrected Ghost');
      subplot(2,1,2); hold on;
      plot(t,abs(Y_actual(:,1)'-g_x_ghost2),'-k');
      title('Absolute Error (m)');
      xlabel('Time (s)'); ylim([0 1]);
end
```

## A.4 Auto-Op Mixer Proof-of-Concept

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:     mixer_thesis.m
% Author:   Graham LeBlanc
%           Dalhousie University
% Email:    graham.leblanc@dal.ca
% Date:     9 Aug., 2011
% Description:
%   This file tries to combine the predictive control with PID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all; clear all; clc;

dt = 0.1; % Needs to be less than vehicle response time
t = 0:dt:20;

% Model params
m = 85;
I = 10; % Moment of inertia
k1 = 40;
k2 = 200;
T_max = 50; % Max thruster force
f_c = [30;0]; % Ocean current force
T_D = 1; % Control time delay [s] (1.5)
T_M = T_D; % Measurement period [s] (≥ dt) (2)

% Update from actual
```

```
i_update = round(T_D/dt)+1;

% PID gains
K_psi_P = 300;
K_psi_D = 200;
K_x_P = 300;            % Manual tuning
K_x_I = 100;
K_x_D = 200;
K_t = sqrt(K_x_I/K_x_D)/K_x_I;  % Tracking (compensator, see Feb. ...
   21 notes)
K = [-K_psi_P -K_psi_D 0 0 0 K_x_P K_x_I K_x_D; ...
    K_psi_P K_psi_D 0 0 0 K_x_P K_x_I K_x_D; ...
    0 0 K_x_P K_x_I K_x_D 0 0 0; ...
    0 0 -K_x_P -K_x_I -K_x_D 0 0 0];
K_rot = [-K_psi_P -K_psi_D 0 0 0 0 0 0; ...
    K_psi_P K_psi_D 0 0 0 0 0 0; ...
    0 0 0 0 0 0 0 0; ...
    0 0 0 0 0 0 0 0];

% Sequence of waypoints [x,y,theta]
P = [5;10;0];
p_i = 1;
p = P(:,p_i); % Current waypoint
r_p = 1; % Waypoint radius
t_th = 0.1; % Heading threshold

% Thruster forces
T = [0;0;0;0];
T_g = T;
T_history = zeros(4,length(t));

% Mixer
e_p_max = 0.5; % Update jump threshold
K_e_start = 0;
K_e_end = 0.5;
K_e = @(e) 0*(e ≤ K_e_start) + ...
    1/(K_e_end-K_e_start)*e*(e > K_e_start && e ≤ K_e_end) + ...
    1*(e > K_e_end);
% K_e = @(e) 1; % Autopilot only
% K_e = @(e) 0; % Predictive only

% Rotation: local -> world
R = @(t) [cos(t) -sin(t); sin(t) cos(t)];

% Force and torque as func of thrusters (local coords)
F = @(T) [0 0 1 -1; 1 1 0 0]*T;
tau = @(T) 0.5*[-1 1 0 0]*T;

% Ocean current force (World -> local)
F_C = @(th) R(th)'*f_c;

% World coords
x = zeros(length(t),2);
y = zeros(length(t),2);
theta = zeros(length(t),2);
x(1,:) = [5 0]; % [pos;spd]
y(1,:) = [0 0];
theta(1,:) = [0 0]; % Heading
x_g = [5 0]; % [pos, spd]
y_g = [0 0];
theta_g = [0 0]; % Heading

% Waypoint history [x,y,theta]
waypoints = zeros(length(t),3);
waypoints(1,:) = [x_g(1) y_g(1) theta_g(1)];

% PID error vector [x;y;theta]
f_error = @(R,x,y,dth,px,py) [R' [0;0];[0 0 1]]*[1 0 -x;0 1 -y; ...
    0 0 dth]*[px;py;1];
previous_error = f_error(R(theta(1)),x(1),y(1), ...
    p(3)-theta(1),p(1),p(2));
e_I = 0; % Integral error
e_out = 0; % commanded-actual output error
previous_error_g = f_error(R(theta_g(1)),x_g(1),y_g(1), ...
    p(3)-theta_g(1),p(1),p(2));
e_I_g = 0; % Integral error
e_out_g = 0; % commanded-actual output error
```

```
% State vectos (local) [pos, spd]
x_s = [0 0];
y_s = [0 0];
% State vectos (local) [pos, spd]
x_s_g = [0 0];
y_s_g = [0 0];

% Graphing
g_X = zeros(length(t),2);
g_X(1,:) = [x(1) y(1)];
g_X_g = zeros(length(t),2);
g_X_g(1,:) = [x_g(1) y_g(1)];
g_V = zeros(length(t),2);
g_V(1,:) = [x(2) y(2)];
g_error_g = zeros(length(t),1);
g_error_g(1) = norm(previous_error_g);
g_theta = zeros(length(t),1);
g_theta(1) = theta(1)+pi/2;
g_T = zeros(length(t),length(T));
g_T(1,:) = T';
g_K = zeros(length(t),1);

% subplot(2,2,[1 3]);
subplot(2,1,1);
hold on;

for i = 2:length(t)

   %% Ghost

   tspan = [t(i-1) t(i)];

    % Update thrusters via PID
    error_g = f_error(R(theta_g(1)),x_g(1),y_g(1), ...
        p(3)-theta_g(1),p(1),p(2));
    e_I_g = e_I_g + (error_g + K_t*e_out_g)*dt; % Windup compensator
    e_D_g = (error_g - previous_error_g)/dt;
    previous_error_g = error_g;
    % If heading is out of threshold, correct first
    if (abs(error_g(3)) > t_th)
        KK = K_rot;
        e_I_g = zeros(length(error_g(:,1)),1);
    else % Otherwise, move normally
        KK = K;
    end

%     KK = K; % Force K
    T = KK*[error_g(3);e_D_g(3); ...
        error_g(1);e_I_g(1);e_D_g(1); ...
        error_g(2);e_I_g(2);e_D_g(2);];

    % Clamp thruster values
    T_C = clamp_thrusters(T,T_max);

    % Save operator command history
    T_history(:,i) = T_C;

    % Get new output error
    e_out_g = [F(T_C);tau(T_C)] - [F(T);tau(T)];

    % Local thruster force--assume const for update period
    F_T = F(T_C); % Thrusters only
    f_t_x = @(t) F_T(1);
    f_t_y = @(t) F_T(2);
    t_t = @(t) tau(T_C);

    % Get local state
    v = R(theta_g(1))'*[x_g(2);y_g(2)]; % Vel: world -> local
    x_s_g = [0 v(1)];
    y_s_g = [0 v(2)];

    % Euler method
    [Tx,X] = tuv_euler(@(ts,xx) ...
        tuv_ode_in(ts,xx,m,k1,k2,f_t_x),tspan,x_s_g);
    [Ty,Y] = tuv_euler(@(ts,yy) ...
        tuv_ode_in(ts,yy,m,k1,k2,f_t_y),tspan,y_s_g);
    [Tw,W] = tuv_euler(@(ts,ww) ...
        tuv_ode_in(ts,ww,m,k1,k2,t_t),tspan,theta_g);

    % Update state (local -> world)
    x_u = [x_g(1);y_g(1)] + R(theta_g(1))*[X(end,1);Y(end,1)]; % Pos
    v_u = R(theta_g(1))*[X(end,2);Y(end,2)]; % Vel

    % State vectors
    x_g = [x_u(1) v_u(1)];
    y_g = [x_u(2) v_u(2)];
    theta_g = W(end,:);

    % If measurement received
    if (i == i_update)

       % Get time span
       n = i-T_D/dt; % Delayed index
       tspan = [t(n) t(i)];



       % Thruster history as func of time
       F_T = F(T_history);
       F_T_x = F_T(1,:);
       F_T_y = F_T(2,:);
       f_t_x = @(t) F_T_x(round(t/dt)+1);
       f_t_y = @(t) F_T_y(round(t/dt)+1);
       t_t = @(t) tau(T_history(:,round(t/dt)+1));

       % Get measured state
       v = R(theta(n,1))'*[x(n,2);y(n,2)]; % Vel: world -> local
       x_s_g = [0 v(1)];
       y_s_g = [0 v(2)];
       t_s_g = theta(n,:);

       % Prediction from delayed state
       [Tx,X] = tuv_euler(@(ts,xx) ...
           tuv_ode_in(ts,xx,m,k1,k2,f_t_x),tspan,x_s_g);
       [Ty,Y] = tuv_euler(@(ts,yy) ...
           tuv_ode_in(ts,yy,m,k1,k2,f_t_y),tspan,y_s_g);
       [Tw,W] = tuv_euler(@(ts,ww) ...
           tuv_ode_in(ts,ww,m,k1,k2,t_t),tspan,t_s_g);

       % Update state (local -> world)
       x_u = [x(n,1);y(n,1)] + R(t_s_g(1))*[X(end,1);Y(end,1)]; % Pos
       v_u = R(t_s_g(1))*[X(end,2);Y(end,2)]; % Vel

       % Updated state
       x_g_u = [x_u(1) v_u(1)];
       y_g_u = [x_u(2) v_u(2)];

       % Only update if in threshold
       if (norm([x_g_u(1)-x_g(1) y_g_u(1)-y_g(1)]) < e_p_max)
         % State vectors
         x_g = x_g_u;
         y_g = y_g_u;
         theta_g = W(end,:);

         e_I_g = zeros(length(error_g(:,1)),1);
       end

       i_update = i_update + round(T_M/dt);

    end

    waypoints(i,:) = [x_g(1) y_g(1) theta_g(1)];

    % Graph
    g_X_g(i,:) = [x_g(1) y_g(1)];


    %% Actual

    if (i > round(T_D/2/dt)+1)

       n = i - round(T_D/2/dt); % Delayed index
       tspan = [t(n-1) t(n)];

       % Prediction error...should WP index be n-1?
       err_p = norm([waypoints(n,1)-x(n-1,1) waypoints(n,2)-y(n-1,1)]);
```

```
        % Update thrusters via PID
        error = f_error(R(theta(n-1,1)),x(n-1,1),y(n-1,1), ...
            waypoints(n,3)-theta(n-1,1),waypoints(n,1),waypoints(n,2));
        e_I = e_I + (error + K_t*e_out)*dt; % Windup compensator
        e_D = (error - previous_error)/dt;
        previous_error = error;

        % If heading is out of threshold, correct first
        if (abs(error(3)) > t_th)
            KK = K_rot;
            e_I = zeros(length(error(:,1)),1);
        else % Otherwise, move normally
            KK = K;
        end

    %     KK = K; % Force K
        T_PID = KK*[error(3);e_D(3); ...
            error(1);e_I(1);e_D(1); ...
            error(2);e_I(2);e_D(2);];

        % Clamp PID
        T_PID_C = clamp_thrusters(T_PID,T_max);

        % Get new output error
        e_out = [F(T_PID_C);tau(T_PID_C)] - [F(T_PID);tau(T_PID)];

        T_g = T_history(:,n-1); % Delayed command

        % Mixed thrusters
        T_C = K_e(err_p)*T_PID_C + (1-K_e(err_p))*T_g;
%         text(x(n-1,1),y(n-1,1),num2str(K_e(err_p)));

        % Local thruster force--assume const for update period
        F_T = F(T_C) + F_C(theta(n-1,1)); % Thrusters + ocean current
        f_t_x = @(t) F_T(1);
        f_t_y = @(t) F_T(2);
        t_t = @(t) tau(T_C);
        % Get local state
        v = R(theta(n-1,1))'*[x(n-1,2);y(n-1,2)]; % Vel: world -> local
        x_s = [0 v(1)];
        y_s = [0 v(2)];
        t_s = theta(n-1,:);

        % Euler method
        [Tx,X] = tuv_euler(@(ts,xx) ...
            tuv_ode_in(ts,xx,m,k1,k2,f_t_x),tspan,x_s);
        [Ty,Y] = tuv_euler(@(ts,yy) ...
            tuv_ode_in(ts,yy,m,k1,k2,f_t_y),tspan,y_s);
        [Tw,W] = tuv_euler(@(ts,ww) ...
            tuv_ode_in(ts,ww,m,k1,k2,t_t),tspan,t_s);

        % Update state (local -> world)
        x_u = [x(n-1,1);y(n-1,1)] + R(t_s(1))*[X(end,1);Y(end,1)]; % Pos
        v_u = R(t_s(1))*[X(end,2);Y(end,2)]; % Vel

        % State vectors
        x(n,:) = [x_u(1) v_u(1)];
        y(n,:) = [x_u(2) v_u(2)];
        theta(n,:) = W(end,:);

        % Graph
        g_X(n,:) = [x_u(1) x_u(2)];
        g_V(n,:) = [v_u(1) v_u(2)];
        g_theta(n) = theta(n,1)+pi/2;
        g_T(n-1,:) = T_C';
        g_error_g(n-1) = norm(error); % Heading as well
        g_K(n,:) = K_e(err_p);

    end

end

% Fill rest of actual array
for n = i - round(T_D/2/dt):i
     g_X(n,:) = g_X(i - round(T_D/2/dt),:);
end
```

```
% Main plot
% subplot(2,2,[1 3]); hold on;
axis equal;
% xlim([3 7]); ylim([0 12]);
xlim([0 12]); ylim([3 8]);
xlabel('x [m]'); ylabel('y [m]');

% Heading
% quiver(g_X(:,1),g_X(:,2),cos(g_theta),sin(g_theta),'g');

% Position
% plot(g_X(:,1),g_X(:,2),'.');
% plot(g_X_g(:,1),g_X_g(:,2),'.r');
% Swap x and y
plot(g_X(:,2),g_X(:,1),'-b');
plot(g_X_g(:,2),g_X_g(:,1),'.r');
title('Command Mixer Response');
xlabel('x [m]'); ylabel('y [m]');
legend('Actual','Ghost');

% Plot waypoints
% plot_waypoints(P,r_p);
% Swap x and y
plot_waypoints([P(2,:);P(1,:);P(3,:)],r_p);

% Plot mix factor
subplot(2,1,2);
plot(t,g_K);
title('Mix Factor');
xlabel('Time [s]');
ylim([0 1]);

% % Thruster force
% subplot(2,2,2);
% plot(t,g_T);
% title('Thruster Force (N)');
% legend('T1','T2','T3','T4');
%
```

```
% % Error
% subplot(2,2,4);
% plot(t,g_error_g,'r');
% title('Error (m)');
```

## A.5 Support Functions

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:     clamp_thrusters.m
% Author:   Graham LeBlanc
%           Dalhousie University
% Email:    graham.leblanc@dal.ca
% Date:     9 Aug., 2011
% Description:
%   Clamps all thrusters to [-T_max, T_max] and scales the rest
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function T_C = clamp_thrusters(T,T_max)

% Find max
TT = max(abs(T));

% Only adjust if over max
if TT > T_max
    T_C = T*T_max/TT;
else
    T_C = T;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:     tuv_euler.m
% Author:   Graham LeBlanc
%           Dalhousie University
% Email:    graham.leblanc@dal.ca
% Date:     9 Aug., 2011
% Description:
%   Euler method performed on given ODE function over span, 'tspan'.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [t,y] = tuv_euler(odefun,tspan,y0)

dt = 0.01; % Calculation time step
t = tspan(1):dt:tspan(end);

% Output array
y = zeros(length(t),length(y0));
y(1,:) = y0;

for i = 2 : length(t)

   % Euler method
   y(i,:) = y(i-1,:) + dt*odefun(t(i-1),y(i-1,:))';

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:     tuv_ode_in.m
% Author:   Graham LeBlanc
%           Dalhousie University
% Email:    graham.leblanc@dal.ca
% Date:     9 Aug., 2011
% Description:
%   ODE function for TUV that accepts a disturbance, 'input'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Input force must be a function of t
function dy = tuv_ode_in(t,y,m,k1,k2,input)

% Construct decoupled DEs
dy = zeros(2,1); % Column vector to hold DEs
dy(1) = y(2);
dy(2) = -k1/m*y(2) - k2/m*y(2)*abs(y(2)) + input(t)/m;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:     plot_waypoints.m
% Author:   Graham LeBlanc
%           Dalhousie University
% Email:    graham.leblanc@dal.ca
% Date:     9 Aug., 2011
% Description:
%   Plots list of waypoints with radius, 'r'.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function plot_waypoints(P,r)

for i = 1:length(P(1,:))

    c = [P(1,i) P(2,i)]; % center
    pos = [c-r 2*r 2*r];
    rectangle('Position', pos, 'Curvature',[r r]);
    scatter(c(1),c(2),'Xk');

end
```

# BIBLIOGRAPHY

[1] D. Blidberg, "The development of autonomous underwater vehicles (auvs); a brief summary," in *IEEE ICRA*, vol. 6500, Citeseer, 2001.

[2] BP, "Remotely Operated Vehicles." BP, 2011. [Online]. Available: `http://www.bp.com/sectiongenericarticle800.do?categoryId=9036600&contentId=7067604` [Accessed: Jul. 2011].

[3] J. Yuh, "Design and control of autonomous underwater robots: A survey," *Autonomous Robots*, vol. 8, no. 1, pp. 7–24, 2000.

[4] Seamore Marine, "SEAMOR 300T." Seamore Marine, 2009. [Online]. Available: `http://www.seamor.com/2009/05/seamor-300t.html` [Accessed: Jun. 2011].

[5] Canadian Scientific Submersible Facility, "ROPOS, Vehicle Specifications." Canadian Scientific Submersible Facility, 2008. [Online]. Available: `http://www.ropos.com/index.php?option=com_content&view=article&id=16&Itemid=20` [Accessed: Jun. 2011].

[6] Oceaneering, "ROV Services Rate Schedule Gulf of Mexico." Oceaneering, 2011. [Online]. Available: `www.oceaneering.com` [Accessed: Feb. 2011].

[7] D. Green and C. Bernstein, "Command, control and data transport for underwater robots using acoustic communications," in *Underwater Technology, 2002. Proceedings of the 2002 International Symposium on*, pp. 343–348, IEEE, Sept. 2002.

[8] E. Marques, J. Pinto, S. Kragelund, P. Dias, L. Madureira, A. Sousa, M. Correia, H. Ferreira, R. Gonçalves, R. Martins, and Others, "AUV control and communication using underwater acoustic networks," in *OCEANS 2007-Europe*, pp. 1–6, IEEE, 2007.

[9] A. P. Aguiar and A. M. Pascoal, "Dynamic positioning and way-point tracking of underactuated AUVs in the presence of ocean currents," *International Journal of Control*, vol. 80, pp. 1092–1108, Apr. 2007.

[10] Monterey Bay Aquarium Research Institute, "Long-range autonomous underwater vehicle Tethys." MBARI, 2010. [Online]. Available: `http://www.mbari.org/auv/LRAUV.htm` [Accessed: Mar. 2011].

[11] P. Beaujean and E. Carlson, "Combined vehicle control, status check and high-resolution acoustic images retrieval using a high-frequency acoustic modem on a hovering AUV," *OCEANS 2010*, 2010.

[12] J. Choi and H. Kondo, "On fault-tolerant control of a hovering AUV with four horizontal and two vertical thrusters," *OCEANS 2010 IEEE-Sydney*, pp. 1–6, 2010.

[13] S.-W. Byun, J.-Y. Kim, S.-K. Lee, J.-C. Park, and K.-S. Kim, “Development and Experiment of a Hovering AUV Tested-bed for Underwater Exploration,” *2007 Symposium on Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies*, pp. 285–291, Apr. 2007.

[14] E. H. Binugroho, T. K. Ha, J. W. Choi, and N. G. Ko, “Modeling of Hovering AUV Testbed and Design of ACE/TAO RTES-Based Open Control Platform,” in *Proceedings of the Nineteenth (2009) International Offshore and Polar Engineering Conference*, vol. 1, (Osaka), 2009.

[15] J. Vaganay, L. Gurfinkel, M. Elkins, D. Jankins, and K. Shurn, “Hovering autonomous underwater vehicle-system design improvements and performance evaluation results,” in *Proceedings of the International Symposium on Unmanned Untethered Submersible Technology (UUST)*, pp. 1–14, 2009.

[16] P. Ballou, “An acoustically controlled tetherless underwater vehicle for installation and maintenance of neutrino detectors in the deep ocean,” tech. rep., Deep Ocean Engineering, Inc., San Leandro, CA, 1997.

[17] M. Stojanovic, “Underwater acoustic communications,” in *Electro/95 International. Professional Program Proceedings.*, pp. 435–440, IEEE, 2002.

[18] D. B. Kilfoyle and A. B. Baggeroer, “The state of the art in underwater acoustic telemetry,” *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, pp. 4–27, 2000.

[19] A. Kaya and S. Yauchi, “An Acoustic Communication System for Subsea Robot,” in *OCEANS’89.*, pp. 765–770, 1989.

[20] M. Stojanovic, “Adaptive multichannel combining and equalization for underwater acoustic communications,” *The Journal of the Acoustical Society of America*, vol. 94, no. 3, p. 1621, 1993.

[21] D. Steinke, “Design and Simulation of a Kalman Filter for ROV Navigation,” Master’s thesis, University of Victoria, 2003.

[22] D. Leader, “Kalman filter estimation of underwater vehicle position and attitude using Doppler velocity aided inertial motion unit,” 1994.

[23] A. Myagotin and I. Burdinsky, “AUV positioning model employing acoustic and visual data processing,” in *OCEANS 2010 IEEE-Sydney*, pp. 1–6, IEEE, 2010.

[24] R. Beckman, A. Martinez, and B. Bourgeois, “AUV positioning using bathymetry matching,” in *OCEANS 2000 MTS/IEEE Conference and Exhibition*, vol. 3, pp. 2123–2127, IEEE, 2000.

[25] F. Dalgleish, S. Tetlow, and R. Allwood, “Experiments in laser-assisted visual sensing for AUV navigation,” *Control engineering practice*, vol. 12, pp. 1561–1573, Dec. 2004.

[26] E. Slawiñski, V. Mut, and J. Postigo, "Teleoperation of mobile robots with time-varying delay," *IEEE TRANSACTIONS ON ROBOTICS*, vol. 23, pp. 1071–1082, Oct. 2006.

[27] T. Fossen, *Guidance and control of ocean vehicles*. John Wiley & Sons, 1994.

[28] A. O'Dwyer, "PID compensation of time delayed processes: a survey," in *Proceedings of the Irish Signals and Systems Conference, Dublin (Ireland)*, pp. 5–12, Citeseer, 2000.

[29] K. Astrom and R. Murray, "PID Conrol," in *Feedback Systems: An Introduction for Scientists and Engineers*, ch. 6, pp. 293–313, Princeton University Press, 2008.

[30] J. Lee, P. Lee, S. Hong, and D. Hong, "Design Of A Digital Control System For Rov," *OCEANS 91 Proceedings*, pp. 1282–1285, 1991.

[31] V. M. Hung and U. J. Na, "Remote control system of a 6 DOF underwater robot," *2008 International Conference on Control, Automation and Systems*, pp. 2575–2580, Oct. 2008.

[32] C. Sayers, R. Paul, L. Whitcomb, and D. Yoerger, "Teleprogramming for subsea teleoperation using acoustic communication," *IEEE Journal of Oceanic Engineering*, vol. 23, no. 1, pp. 60–71, 1998.

[33] J. Guo, F. Chiu, S. Cheng, and J. Pan, "Robust trajectory control of a remotely operated vehicle for underwater inspection tasks," in *Scientific Use of Submarine Cables and Related Technologies, 2003. The 3rd International Workshop on*, pp. 161–165, IEEE, 2003.

[34] C. Caldwell, E. Collins, and S. Palanki, "Integrated Guidance and Control of AUVs Using Shrinking Horizon Model Predictive Control," *Oceans 2006*, vol. 3, pp. 1–6, Sept. 2006.

[35] W. Naeem, R. Sutton, J. Chudley, F. R. Dalgleish, and S. Tetlow, "A genetic algorithm-based model predictive control autopilot design and its implementation in an autonomous underwater vehicle," *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, vol. 218, pp. 175–188, Jan. 2004.

[36] D. Yoerger and J. Slotine, "Robust trajectory control of underwater vehicles," *IEEE Journal of Oceanic Engineering*, vol. 10, pp. 462–470, Oct. 1985.

[37] T. Chatchanayuenyong and M. Parnichkun, "Time Optimal Hybrid Sliding Mode-PI Control for an Autonomous Underwater Robot," *International Journal of Advanced Robotic Systems*, vol. 5, no. 1, pp. 91–98, 2008.

[38] L. R. Young, "Human Control Capabilities," in *Biostronautics Data Book* (Parker and West, ed.), ch. 16, pp. 751–806, NASA, 2nd ed., 1973.

[39] C. Bulich, a. Klein, R. Watson, and C. Kitts, "Characterization of delay-induced piloting instability for the Triton undersea robot," *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No.04TH8720)*, pp. 409–423, 2004.

[40] C. Cheng, *Predictor displays: Theory development and application to towed submersibles*. PhD thesis, Massachusetts Institute of Technology, 1991.

[41] J. Funda and R. Paul, "Efficient control of a robotic system for time-delayed environments," in *Advanced Robotics*, pp. 219–224, IEEE, 1991.

[42] C. S. Edrington, "Continuous-Time Control Systems: Root-Locus Analysis." University of Texas, 2006. [Online]. Available: `http://www.ece.utexas.edu/~friedric/clas11.pdf` [Accessed: Jun. 2011].

[43] G. A. Baker and P. Graves-Morris, *Padé Approximants (Encyclopedia of Mathematics and its Applications)*. Cambridge University Press, 1996.

[44] T. Sheridan, "Space teleoperation through time delay: Review and prognosis," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 592–606, 2002.

[45] M. Caccia, G. Indiveri, and G. Veruggio, "Modeling and identification of open-frame variable configuration unmanned underwater vehicles," *IEEE Journal of Oceanic Engineering*, vol. 25, pp. 227–240, Apr. 2000.

[46] Y. Eng, W. Lau, E. Low, G. Seet, and C. CS, "Estimation of the Hydrodynamics Coefficients of an ROV using Free Decay Pendulum Motion," *Engineering Letters*, vol. 16, no. 3, 2009.

[47] G. LeBlanc, P. Gregson, and J. Gu, "A Control Continuum for Tetherless Underwater Vehicles," in *24th Canadian Conference on Electrical and Computer Engineering*, 2011.

[48] A. Bejczy, W. Kim, and S. Venema, "The phantom robot: predictive displays for teleoperation with time delay," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pp. 546–551, IEEE, 1990.

[49] Marine Simulation LLC, "ROVsim Pro." Marine Simulation LLC, 2011. [Online]. Available: `http://marinesimulation.com/?page_id=35` [Accessed: Jun. 2011].

[50] T. Bräunl, "SubSim - An Autonomous Submarine Simulation System." The University of Western Australia - Robotics & Automation Lab, 2006. [Online]. Available: `http://robotics.ee.uwa.edu.au/auv/subsim.html` [Accessed: Jun. 2011].

[51] Microsoft, "Microsoft Robotics Developer Studio." Microsoft Robotics, 2011. [Online]. Available: `http://www.microsoft.com/robotics/` [Accessed: Jul. 2011].