Autonomous Control and Simulation of the VideoRay Pro III Vehicle Using MOOS and IvP Helm

Anthony Spears
Department of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA, USA

Michael West, Thomas Collins
Electronic Systems Laboratory (ELSYS)
Georgia Tech Research Institute
Atlanta, GA, USA

***Abstract*—Most underwater vehicles are controlled by human operators through tethered cables which inhibit range and affect the hydrodynamics. The overall performance of these promising vehicles would improve through the employment of higher levels of autonomy. Implementation of open source autonomous guidance software in an off the shelf underwater vehicle is explored here as a solution. Development and implementation of this autonomous guidance and vehicle control is greatly facilitated through the use of an accurate vehicle simulator. Running real world tests of an underwater vehicle is extremely time intensive and error prone. The analysis of the vehicle performance underwater is extremely challenging with limited accurate positioning sensing e.g. GPS. A vehicle simulator allows for faster development of the system by providing vehicle performance information prior to expensive real world missions. This research presents a method for simulation and testing of autonomous guidance and control in a vehicle accurate simulator for the VideoRay Pro III underwater vehicle and demonstrates the capability through simulated examples and analysis.**

Keywords—UUV; AUV; MOOS; IvP; PID controller; guidance; control; unmanned; simulation; hydrodynamics; thruster; kinematics; VideoRay

I. Introduction

Unmanned underwater vehicles (UUV's) have become increasingly popular in recent years as the technology has advanced and the development costs have been driven down. As the level of autonomy increases, UUVs have provided new capabilities and reduced reliance on human operators. In this research, a commercial remotely operated vehicle (ROV), the VideoRay Pro III, is considered for the development of a fully autonomous vehicle whose end goal is the detection and tracking of underwater cables and other man-made objects.

The research will present the development of a simulation environment along with the vehicle modeling. This simulated environment will allow the dynamics of the system to be simulated prior to the actual mission. The hydrodynamics of the vehicle are modeled as well as the low level control of the thrusters on the vehicle, resulting in a simulation that is very accurate to actual vehicle performance. The VideoRay model and simulator are able to effectively simulate the effects of vehicle thrusters and external forces on the AUV. The final output of the simulator provides full state information including speed and position.

The higher level guidance decisions are made using IvP Helm [3], a backseat driver autonomy engine that is part of MOOS (Mission Oriented Operating Suite). MOOS [10] is a publish-subscribe marine middleware operating system similar to ROS [9] (Robot Operating System), the very popular middleware for ground robotics, that facilitates simple communication and modularity between applications. IvP Helm will be used on the final vehicle as well as the vehicle simulator for guidance control. Another MOOS application called pMarineViewer is used for a graphical 2D simulation of the vehicle motion. The example simulation presented will illustrate the vehicle simulator and IvP Helm with an exploration of the floor of the Charles River. In the physical implementation of this system, the VideoRay simulator MOOS application will not be used, and instead will be replaced with sensors and MOOS applications needed for determination of speed, location and orientation inside the actual UUV.

II. Background and Literature Survey

As mentioned previously, the framework and environment used for this modeling and simulation project is a middleware operating system called MOOS. MOOS was written by Paul Newman in 2001 to support operations with autonomous marine vehicles at MIT [3]. MOOS forms open source middleware (glue that connects a collection of applications) for the development of autonomous marine vehicle systems. The heart of MOOS is the MOOSDB, which is a publish-subscribe database for inter-process communication. Publications are messages sent from a process, while subscriptions are the messages received from another process. If an application wants to receive messages from another process, it must subscribe to the desired variable with a desired update rate. The MOOS process reads configuration parameters on startup from a mission file (*.moos), which can be shared by multiple MOOS processes.

A MOOS application called MOOS-IvP Helm was developed in 2004 at MIT for autonomous control of

978-0-933957-40-4 ©2013 MTS

unmanned marine surface craft, and was later extended for use in underwater platforms [3]. It uses multi-objective optimization to implement autonomous behavior coordination in the platform using interval programming. IvP uses a backseat driver paradigm (vehicle control is separate from the autonomy), and provides behavior based autonomy. The IvP solver publishes the single best vehicle action to the MOOSDB based on competing IvP behaviors produced from navigation and state information. The configuration of the IvP helm is completed with a *.bhv behavior file, which provides mission configuration of the helm behaviors. The IvP application subscribes to sensor information and other high level command and control that it needs to make decisions. The Helm provides autonomous guidance for the vehicle based on the configuration in the corresponding behavior file.

The vehicle model used in this research is based off of the work done by Wei Wang in his thesis [1] at the University of Waterloo. Wang was able to obtain the added mass coefficients, moments of inertia, and linear and quadratic drag terms for the VideoRay Pro III through a method called strip theory [6] as well as experimentation. The model developed is based on the underwater vehicle model theory presented by Fossen [2]. Wang also considers simplifications to the vehicle dynamics such as zero roll and pitch at all time, which is reasonable to assume in this UUV, and greatly simplifies the dynamic vehicle model equations of motion. The nonlinear thruster equations relating thrust and thruster propeller speed are also derived and verified by Wang. Wang provided significant background for the simulation and modeling aspect of the project undertaken here.

From the University of Zagreb, an implementation of Wang's original model was attempted using MOOS [5]. Unfortunately, the work was not entirely implemented. However, it provided the framework for the development of this research. In [5], the four MOOS modules used are Destination Control, Control, Propulsion, and AUV (autonomous underwater vehicle). The Destination Control module allows for user input into a terminal window of X, Y, Z coordinates that are then published as the new destination of the vehicle. The Control module subscribes to the destination coordinates as well as the current state of the vehicle and publishes the required thruster propeller speeds for each of the three thrusters needed to make progress toward the destination. The control method used in [1] and [5] is the integrator back-stepping technique to create a Lyapunov stable tracking controller. These thruster control values are subscribed to by the Propulsion module that calculates the external forces in the x, y, z, and ψ directions resulting from these control values. This vector of force and torque values is then published to the AUV module which updates the state of the vehicle based on how the external forces changed the dynamics of the system during the previous time step. Unfortunately, when using the Zagreb implementation, the system becomes unstable responding to the second destination after the first has been sent. Additionally, a second problem is encountered when the MOOS IvP Helm application sends desired speed, desired heading, and desired depth instead of desired destination coordinates. In order to eliminate these problems, new control, propulsion, and AUV modules were developed.

III. System Overview

The system developed for autonomous control and simulation of the VideoRay Pro III UUV consists of a high level guidance sub-system, a low level PID controller for the three controllable degrees of freedom on the vehicle, a module to convert the desired forces and torques to thruster propeller speeds, and a vehicle model dynamics simulator. An overview of the interactions of all of these modules is illustrated in Fig. 1, with the vehicle modeling modules inside of the dashed line and the IvP Helm modules outside the dashed line. The publish and subscribe messages of each module are listed by the arrows between the modules. Together, these sub-systems provide means of autonomously controlling a simulated vehicle to run a mission and yield a very close approximation to the real-world behavior of the VideoRay Pro III UUV.

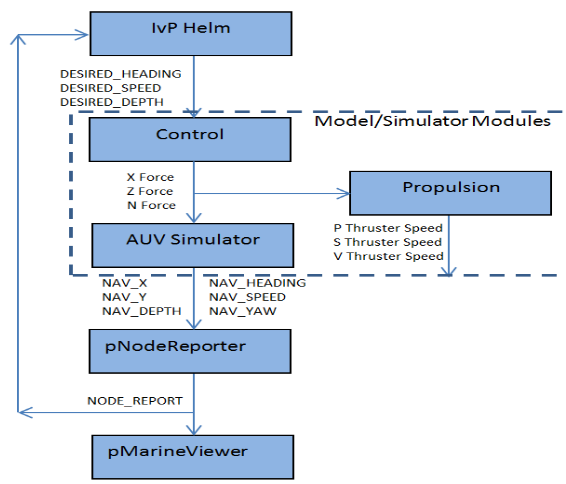

Fig. 1. System module interaction and communication

The MOOS database and operating system is configured using a mission file which launches and configures all applications needed for the system to run. In this case the applications launched are the MOOSDB for system communication, pLogger for error and communication logging, pNodeReporter, pHelmIvP, pMarineViewer, Control, Propulsion, and AUV, the final six of which encompass the vehicle control and simulation modules. The simulation is configured to run at five times the normal speed using "MOOSTimeWarp = 5" in the mission configuration file. The latitude and longitude of the origin of the vehicle in the simulation are configured to be the Charles River by setting *LatOrigin* and *LongOrigin* in the mission configuration file. All of the model and control parameters for the UUV are set in the mission file as well, and are read into the corresponding modules at run time.

The high level guidance of the vehicle is provided by the IvP Helm MOOS application. This module is configured in the corresponding behavior file, which allows for the selection of a path for the vehicle to follow, what to do when it is finished or commanded to return, and when to begin following the prescribed path. In the example presented here, the Helm is configured to guide the vehicle through a lawnmower pattern when the vehicle is commanded to DEPLOY, and configured to return the vehicle to the base point and station keep when it is finished or when it is commanded to RETURN. The depth guidance is configured to keep the vehicle at a depth of 10 meters while following the path and return to the surface when at or returning to the base point. The IvP Helm application publishes desired speed, desired heading, and desired depth to the MOOSDB and uses the vehicle state (x coordinate, y coordinate, depth, heading, speed) to calculate new guidance control values.

The desired speed, heading and depth parameters calculated from the guidance module are used by the Control module, which simulates the low level controllers on the UUV. Using the current state vector obtained from the vehicle simulator AUV module, the error between the desired and actual speed, heading and depth are calculated. This error is used in the PID controllers to calculate the desired X, Z, and N external forces on the vehicle. These forces correspond to forces in the x direction, z direction and torque around the z-axis respectively. In the actual VideoRay Pro III vehicle the depth, speed and heading can be controlled independently using constant depth, speed and heading commands sent to the vehicle [7] [8]. The Control module simulates these three low level controllers on the VideoRay vehicle and should yield fairly accurate simulations of how the vehicle behaves in real missions.

The Propulsion module converts the desired forces and torques to control values for the thrusters on the vehicle. When implemented in the actual vehicle, these control values would be used to control the motors in each of the thrusters. There is a thruster on the top of the VideoRay Pro III UUV which controls the depth of the vehicle, as well as a port thruster and a starboard thruster to control the forward motion and rotation of the UUV. These thrusters are controlled independently and the forward motion is controlled with the sum of the port and starboard thrusters while the rotation is controlled by the difference between them.

The desired forces and torques (X, Z, N) are also used to update the vehicle state in the vehicle dynamics modeling module AUV. The VideoRay Pro III model is used to obtain the updated state of the vehicle after the application of forces from the thrusters at each time step. The new state of the vehicle is published for use in calculation of the next guidance and control values.

At the same time, selected state updates are published by the AUV module for consumption by the pNodeReporter module. The pNodeReporter module is a tool used with the IvP Helm application, which subscribes to the vehicle's current x coordinate, y coordinate, speed, heading, and depth. This module combines all of this state information into one string variable called a NODE_REPORT and publishes the report for consumption by the IvP Helm as well as the pMarineViewer.

The pMarineViewer module subscribes to the NODE_REPORT variable and presents the user with a 2D representation of the vehicle movements over time [4]. The simulation of the vehicle's movements is presented in a window as shown in Fig. 2 with a map or satellite image as a background and the vehicle represented by a small icon. Upon receiving new state information about the vehicle, the pMarineViewer updates the location and orientation of the UUV icon on the map. Other information about the vehicle such as depth and time since last communication is also given in text boxes in the pMarineViewer window.

Fig. 2 depicts the pMarineViewer representation of the example mission used in this research. Here the UUV is in search mode following the lawnmower pattern that is shown as lines in Fig. 2 at a depth of 10 meters. Fig. 3 represents the same mission with the UUV in station-keep mode after it has been commanded to return to the dock and a depth of zero meters. The pMarineViewer application is configured in the MOOS mission file. The satellite image is configured to be an image of the Charles River and the origin is set at the launch dock, corresponding to the origin latitude and longitude set earlier. The Deploy and Return buttons are configured to send signals to the IvP Helm to provide the user high level control of the vehicle during the mission. The pMarineViewer application provides a useful and intuitive method for viewing the progress of the UUV during the missions.

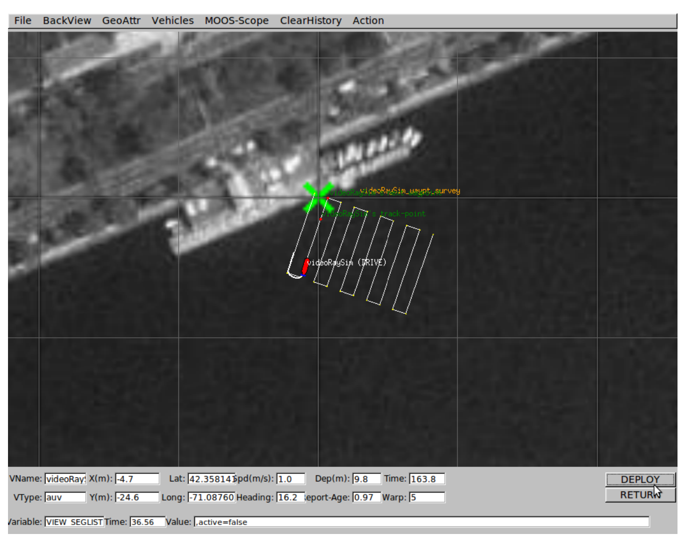

Fig. 2. The pMarineViewer application showing the UUV running the example search mission

Fig. 3. The pMarineViewer application showing the UUV returning to the dock in the example search mission

IV. Matlab Simulation

Prior to implementation in MOOS, the UUV control and vehicle simulation was implemented using Matlab. The modules simulated in Matlab consist of the Control, Propulsion and AUV modules. This allowed for the response of the vehicle to be analyzed graphically and facilitated debugging of errors and tuning of the PID controllers. The example situation run in the Matlab simulation involved the vehicle being commanded to a depth of 10 meters, a speed of one m/s and a ψ of π at time t=10 seconds and then all back to zero at time t=100 seconds. The response of important state and control variables produced from this Matlab simulation can be seen throughout this paper. The responses correlate well with what is seen in the MOOS vehicle simulations.

V. Vehicle Hydrodynamic Model Equations

One of the standard methods for modeling an underwater vehicle uses equations derived in [2]. Using this method, two coordinate frames are defined. The body-fixed reference frame is used for describing the linear and angular velocity while the earth-fixed reference frame is used for describing the position and orientation of the vehicle. The position and orientation of the vehicle is calculated relative to the earth-fixed frame origin. Table 1 from [2] defines the notation used for each of the degrees of freedom. The six degrees of freedom can be summarized with the three vectors in equations 5.1-5.3. η in equation (5.1) represents the position and orientation vector in the earth-fixed frame. ν in equation (5.2) represents the linear and angular velocity vector in the body-fixed frame. T in equation (5.3) represents the external forces and torques acting on the vehicle in the body-fixed frame.

$$\boldsymbol{\eta} = [x, y, z, \phi, \theta, \psi]^T \tag{5.1}$$

$$\boldsymbol{\nu} = [u, v, w, p, q, r]^T \tag{5.2}$$

$$\mathbf{T} = [X, Y, Z, K, M, N]^T \tag{5.3}$$

TABLE I. NOTATION USED FOR MARINE VEHICLES FROM [2]

DOF		Forces and Moments	Linear and Angular Velocities	Position and Euler Angles
1	Motions in the x-direction (surge)	X	u	x
2	Motions in the y-direction (sway)	Y	v	y
3	Motions in the z-direction (heave)	Z	w	z
4	Rotation about the x-axis (roll)	K	p	Φ
5	Rotation about the y-axis (pitch)	M	q	θ
6	Rotation about the z-axis (yaw)	N	r	ψ

The relation between the earth-fixed and body-fixed frame for the VideoRay can be seen in Fig. 4 from [1].

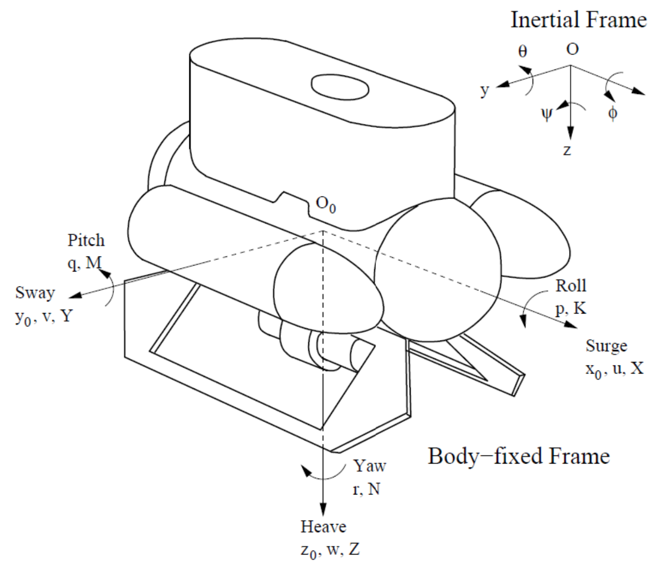

Fig. 4. Earth-fixed and body-fixed frames for the VideoRay UUV [1]

The transformation matrix is of the form shown in Fig. 5. To transform linear and angular velocity from the body-fixed frame to the earth-fixed frame, the transformation matrix $\boldsymbol{J}(\boldsymbol{\eta})$ is used in equation (5.4) representing the kinematic equations in vector form.

$$\dot{\boldsymbol{\eta}} = \mathbf{J}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{5.4}$$

$$\begin{bmatrix} \cos(\psi)\cos(\theta) & \cos(\psi)\sin(\phi)\sin(\theta) - \cos(\phi)\sin(\psi) & \sin(\phi)\sin(\psi) + \cos(\phi)\cos(\psi)\sin(\theta) & \begin{matrix} 0 & 0 & 0 \end{matrix} \\ \cos(\theta)\sin(\psi) & \cos(\phi)\cos(\psi) + \sin(\phi)\sin(\psi)\sin(\theta) & \cos(\phi)\sin(\psi)\sin(\theta) - \cos(\psi)\sin(\phi) & \begin{matrix} 0 & 0 & 0 \end{matrix} \\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\phi)\cos(\theta) & \begin{matrix} 0 & 0 & 0 \end{matrix} \\ & \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} & & \begin{matrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{matrix} \end{bmatrix}$$

Fig. 5. The Body-frame, fixed-frame transformation matrix

The overall dynamic equations of motion for a UUV are represented in [2] in the form of equation (5.5). In this equation **M** represents the vehicle inertia matrix (including added mass), **C** represents the matrix of Coriolis and centripetal terms (including added mass), **D** represents the damping matrix, **g** represents the vector of gravitational forces and moments, and $\boldsymbol{\tau}$ represents the vector of control inputs.

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} \qquad (5.5)$$

In this case of this paper, as in [1], the body-fixed frame is coincident with the three principal axes of inertia, resulting in the inertia matrix reducing to:

$$\begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}$$

where I_x, I_y, I_z are the moments of inertia about the x, y, and z axis respectively. The symmetry of the UUV allows the center of buoyancy of the UUV to be chosen coincident with the origin of the earth-fixed frame, resulting in the center of buoyancy reducing to $r_B = [0\ 0\ 0]^T$. In a similar manner, because of the symmetry of the vehicle and the choice of the earth-fixed frame origin, the center of gravity has null x and y components, so $r_G = [0\ 0\ z_G]^T$. From [1], the weight and buoyancy distribution of the VideoRay UUV will enforce a constant zero pitch and roll state, so it can be assumed that for all time $\Phi = \theta = p = q = 0$. This means that the thrusters on the UUV only have effect on surge, heave, and yaw motion [1], leaving only X, Z and N in the **T** force/moment vector. This also suggests that the system can be decomposed into two independent systems in the vertical and horizontal planes [1]. One further assumption made here is that the vehicle is always neutrally buoyant (B = W). With these assumptions, the dynamic equations of motion are reduced to equations (5.6)-(5.9).

$$(\mathrm{m} - \mathrm{X}_{\dot{u}})\dot{\mathrm{u}} = -(\mathrm{Y}_{\dot{v}} - \mathrm{m})\mathrm{vr} + \mathrm{X_u u} + \mathrm{X}_{\mathrm{u|u|}}\mathrm{u|u|} + \mathrm{X} \qquad (5.6)$$

$$(\mathrm{m} - \mathrm{Y}_{\dot{v}})\dot{\mathrm{v}} = (\mathrm{X}_{\dot{u}} - \mathrm{m})\mathrm{ur} + \mathrm{Y_v v} + \mathrm{Y}_{\mathrm{v|v|}}\mathrm{v|v|} \qquad (5.7)$$

$$(\mathrm{m} - \mathrm{Z}_{\dot{w}})\dot{\mathrm{w}} = \mathrm{Z_w w} + \mathrm{Z}_{\mathrm{w|w|}}\mathrm{w|w|} + \mathrm{Z} \qquad (5.8)$$

$$(\mathrm{I_z} - \mathrm{N}_{\dot{r}})\dot{\mathrm{r}} = \mathrm{N_r r} + \mathrm{N}_{\mathrm{r|r|}}\mathrm{r|r|} + \mathrm{N} \qquad (5.9)$$

In equations (5.6)-(5.9) above:

- m represents the mass of the vehicle in kg
- $X_{\dot{u}},\ Y_{\dot{v}},\ Z_{\dot{w}},\ N_{\dot{r}}$ represent the added mass constants for the vehicle
- $X_u,\ Y_v,\ Z_w,\ N_r$ represent the linear drag hydrodynamic coefficients
- $X_{u|u|},\ Y_{v|v|},\ Z_{w|w|},\ N_{r|r|}$ represent the quadratic drag hydrodynamic coefficients

There is a term $[-(X_{\dot{u}} - Y_{\dot{v}})uv]$ on the right hand side of equation 5.9 for $\dot{r}$ that is assumed to be zero in [1]. There is a term [-(B - W)] on the right hand side of the third equation for $\dot{w}$ that is zeroed as well due to the neutral buoyancy assumption. With these simplifications, the z_G, I_x, and I_y terms are eliminated and no longer needed. The vehicle modeling constants such as added mass, drag hydrodynamic coefficients, and moments of inertia are determined in [1] using strip theory. Strip theory involves a method of dividing the vehicle into a number of strips and computing the constants for each 2D strip, then summing over the length of the vehicle body [6]. This method provided the constants in Table 2.

TABLE II. ADDED MASS, DRAG COEFFICIENTS FOR THE UUV MODEL

Added Mass Coefficients	**Linear Drag Coefficients**	**Quadratic Drag Coefficients**
$X_{\dot{u}} = 1.94$	$X_u = -2.3$	$X_{u\|u\|} = -8.28$
$Y_{\dot{v}} = 6.05$	$Y_v = -8.01$	$Y_{v\|v\|} = -23.69$
$Z_{\dot{w}} = 3.95$	$Z_w = -5.81$	$Z_{w\|w\|} = -20.52$
$N_{\dot{r}} = 1.18\text{e-}2$	$N_r = -0.0048$	$N_{r\|r\|} = -0.0089$

The linear and quadratic coefficients are negative because drag induces a force in the opposite direction of the vehicle motion. The N_r constant is the only term that uses an experimental value instead of the analytical value determined from strip theory. If the analytical value was used, the $(I_z - N_{\dot{r}})$ term in the $\dot{r}$ motion equation would be negative. This would result in a sign flip propagating through the entire equation, as I_z is smaller in magnitude than the analytical value obtained for $N_{\dot{r}}$. The other constant values used in the UUV dynamic equations of motion are:

- m = 15 – The mass of vehicle – approximately 15kg
- g = 9.8 – The gravity coefficient
- Iz = 0.0253 – The moment of inertia about the Z axis
- R = 0.175 – The distance from the origin of the vehicle to the horizontal thrusters (m)
- Time Step = 0.1 – The time step between state updates – 100ms is used in MOOS (10Hz)

VI. VEHICLE THRUSTERS

Once the desired external forces and torques (X, Z, N) are obtained using the PID controllers, these values are saturated at their maximum and minimum values and then translated into thruster propeller speeds. These speed values are used as inputs to directly control the thrusters. As determined in [1] the max thruster propeller speed is 150/s and the minimum thruster propeller speed is -150/s. These are the limits placed on the control values *n*. The thruster coefficients relating the thruster propeller speed with the thrust output in newtons are shown in Table 3.

TABLE III. THRUSTER COEFFICIENTS

	Forward	Backward
Horizontal (Port/Starboard)	Cthf = 2.59 x 10^{-4}	Cthb = 1.01 x 10^{-4}
Vertical	Ctvf = 1.19 x 10^{-4}	Ctvb = 7.53 x 10^{-5}

The nonlinear equation for thrust T as derived in [1] is shown in equation (6.1) in terms of the control value n as well as the thruster constant C_t.

$$T = C_t n|n| \tag{6.1}$$

To obtain torque from thrust values, one must multiply by the distance to the axis of rotation, which in this case is R. The equation for the ψ torque is then $Torque = T * R$ where T is thrust in Newtons, n is the control value corresponding to thruster propeller speed, C_t is the thruster coefficient from Table 3, and R is the distance from the thruster to the center of the vehicle. To obtain a formula for the control value (n) in terms of the desired thrust (T), the positive n and negative n cases are considered separately to eliminate the problems with the absolute value in equation 6.1. The sign of n and T will always be the same for each thruster. The two equations used to obtain the control value (n) corresponding to thruster propeller speed are shown in equations (6.2) and (6.3).

TABLE IV. THRUSTER CONTROL PROPELLER SPEED EQUATIONS

Case when both n and T are negative	Case when both n and T are positive or zero
$n = -\sqrt{-\frac{T}{C_t}}$ (6.2)	$n = \sqrt{\frac{T}{C_t}}$ (6.3)

The thrust and torque values (X, Z, N) come from the PID controllers and correspond to desired external forces and torque to apply to the vehicle. The X thrust is a summation of the thrust from the starboard thruster and the thrust from the port thruster. The Z thrust is a direct mapping of the thrust from the vertical thruster. The N thrust is the summation of the thrust from the starboard thruster and the negative thrust from the port thruster, or the difference between the starboard thruster and the port thruster, and must be multiplied by R to obtain torque instead of thrust or force. These relations are summarized in Table 5. The desired X, Z, and N force and torque values as well as the corresponding responses of the three thrusters to the example commands described in the Matlab Simulation section of this paper can be seen in Fig. 6-10.

TABLE V. EQUATIONS FOR THE DESIRED THRUST AND TORQUE VALUES

X (Desired forward thrust)	Z (Desired vertical thrust)	N (Desired z-axis torque)
$X = T_S + T_P$	$Z = T_V$	$N = R*(T_S - T_P)$

Combining and substituting the equations for X and N yields the equations (6.4) and (6.5) for T_S and T_P. These equations represent the starboard and port thrust (T_S and T_P) in terms of desired forward thrust (X), desired torque around the z-axis (N) and distance from the origin to the thrusters (R).

$$T_S = \frac{X + (\frac{N}{R})}{2} \tag{6.4}$$

$$T_P = \frac{X - (\frac{N}{R})}{2} \tag{6.5}$$

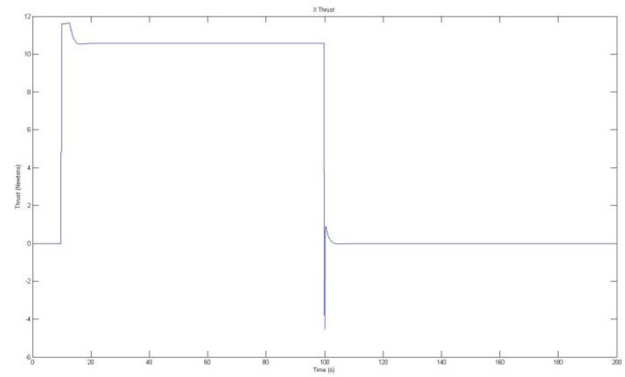

Fig. 6. Thrust or force applied to the vehicle in the forward direction by the two horizontal thrusters

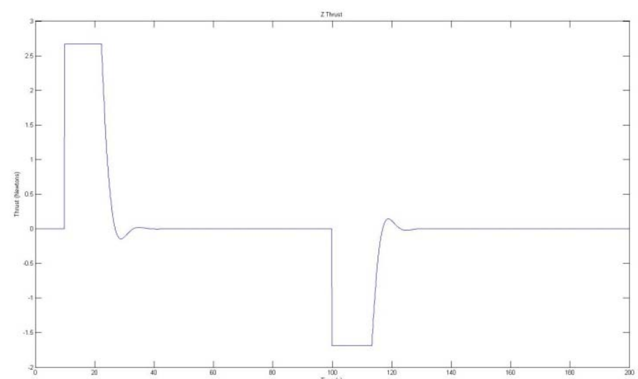

Fig. 7. Thrust or force applied to the vehicle in the vertical direction by the vertical thruster

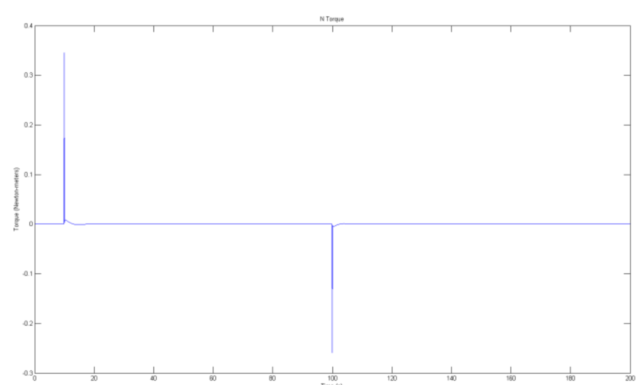

Fig. 8. Torque applied to the vehicle in the psi direction around the z-axis by the two horizontal thrusters

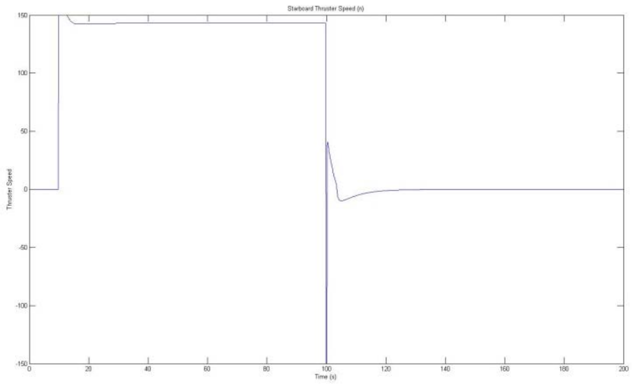

Fig. 9. The Control value for the starboard thruster, representing speed of the thruster propeller. This is very similar to the response of the port thruster, so it is omitted.

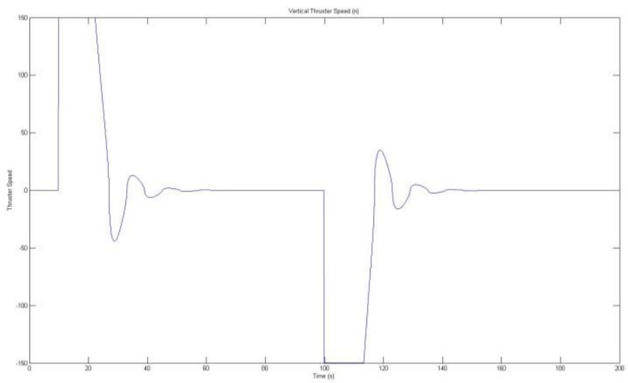

Fig. 10. Control value for the vertical thruster, representing speed of the thruster propeller

In order to ensure the control values (*n*) sent to the thrusters stay in the limit of (-150, 150), the desired thrust and toque values must be saturated at their maximum and minimum values corresponding to these maximum and minimum control values. To obtain these values, the maximum and minimum control values (+/-150) are plugged into equation 6.1, yielding the results in Table 6.

TABLE VI. THRUST LIMITS FOR EACH THRUSTER

	Minimum Thrust	Maximum Thrust
Horizontal (Port/Starboard) Thruster	-2.27 newtons	5.82 newtons
Vertical Thruster	-1.69 newtons	2.67 newtons

To find the limits for the desired thrust and torque values (X, Z, N) we use the equations in Table 5 and the maximum and minimum thruster thrust values in Table 6, summarized in Table 7.

TABLE VII. THRUST LIMITS FOR THE DESIRED THRUST AND TORQUE

	Minimum Thrust/Torque	Maximum Thrust/Torque
X	2*(Min horizontal thrust) = -4.54 newtons	2*(Max horizontal thrust) = 11.64 newtons
Z	Min vertical thrust = -1.69 newtons	Max vertical thrust = 2.67 newtons
N	R*[(Min horizontal thrust) – (Max horizontal thrust)] = -1.416 newton-meters	R*[(Max horizontal thrust) – (Min horizontal thrust)] = 1.416 newton-meters

At this point, constraints are added to the desired thrust and torque calculations. There is a dependency between forward force (X) and z-axis torque (N). They both have the two horizontal thrusters (port and starboard) as components in their overall calculation. To resolve these dependency issues, the yaw torque calculation is considered first, followed by the forward thrust calculation, which assigns the yaw torque the higher importance. Considering the N calculation independently, it can be seen that the limit on the possible N torque that can be applied while keeping the forward thrust (X) zero would be the case when 2.27 newtons are applied with opposite sign to the two horizontal thrusters. This is limited by the backward horizontal thrust limit in Table 6. This implies that the maximum and minimum N torque values are actually +/- 0.7945 newton-meters. Once the desired N value is limited to this constraint, the desired forward thrust can be added in (constrained by the limits in Table 6) to obtain the final thrust values for horizontal thrusters.

These constraints and calculations can be thought of as a sliding bar with a length of (N/R) and an absolute middle value of X/2. The top and bottom of the bar are then equal to the thrust values of the two horizontal thrusters (Tp and Ts). The bar hits a ceiling and floor at the limits of the horizontal thrusters shown in Table 6. Using this method, the width of the bar is calculated (N/R) and then can be slid up and down to obtain the X value without violating any thruster limits. The controller in the end system calculates desired N torque and constrains it to the limits of N as in Table 7. Desired X is then added as long as it won't violate the max thrust for either the port or starboard thrusters. If the maximum or minimum limit is reached, the desired X is scaled down to the point where the thruster is saturated as this maximum or minimum respectively, causing one of the thrusters to be operating at a maximum or minimum speed, but preserving the desired N torque. A demonstration of the constraints placed on the X force and N torque is shown in Fig. 11. The far left shows how the method for choosing these two control values begins by setting the desired N value which sets the size of the bar. The second from the left shows that once the N value is chosen, the bar is slid up or down to add in the X force component. The third from the left shows an example of a desired thrust and torque that is outside of the limits of the thrusters. The far right shows how this problem is solved by reducing the X value, which shifts the bar down into the allowed limits, while preserving the value of N (the size of the bar). The Matlab simulation code enforcing these constraints is given in Fig. 12.

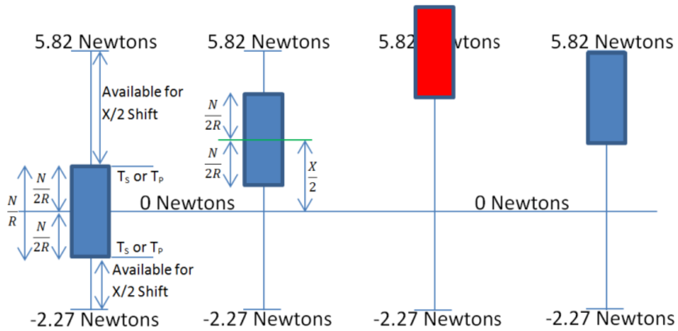

Fig. 11. X force and N torque selection demonstration

```
if N > MaxN
      N = MaxN;
elseif N < MinN
      N = MinN;
end

if X > (MaxX - (abs(N)/R))
      X = (MaxX - (abs(N)/R));
elseif X < (MinX + (abs(N)/R))
      X = (MinX + (abs(N)/R));
end
```

Fig. 12. Matlab code enforcing the constraints on X and N desired force and torque. The desired N torque is calculated first, so it is given priority.

VII. Coordinate System Translations

There is a discrepancy between the coordinate systems of the MOOS IvP Helm environment and the vehicle model and simulation environment in [2]. To move from one to the other, translation equations must be developed. The only variable that is affected in this system is the psi (ψ), or yaw variable which in MOOS IvP Helm is equivalent to the heading variable. In the vehicle model and simulation environment, the psi variable is in radians and its value is constrained to [0:2π]. In IvP Helm the heading variable is in units of degrees and has values constrained to [0:359]. In the vehicle and model simulation environment psi is equal to zero when the vehicle is pointing in the direction of the positive x-axis, increasing in a positive fashion as it moves toward pointing in the positive y-axis direction. In the IvP Helm environment, heading is equal to zero when the vehicle is pointing in the direction of the positive y-axis, increasing in a positive fashion as it moves toward pointing in the positive x-axis direction. The coordinate systems are flipped in sign so that the z axis goes into the water for the simulation environment and comes out of the water in the IvP Helm environment. There is also a translation of 90 degrees between the two coordinate systems. The two coordinate systems are displayed below in Fig. 13 and 14. The equations to translate between these two coordinate systems are shown in equations 7.1 and 7.2.

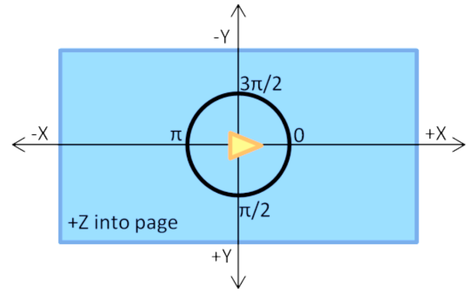

Fig. 13. Overhead view of the vehicle modeling, simulation coordinate system

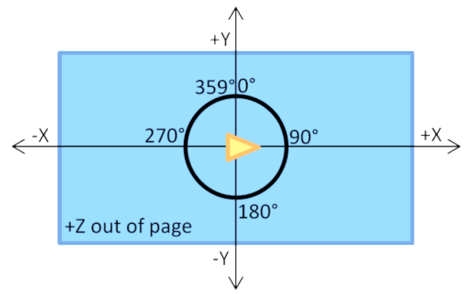

Fig. 14. Overhead view of the MOOS Helm IvP coordinate system

$$\psi = -\frac{\text{Heading}}{180} * \pi + \frac{\pi}{2} \qquad (7.1)$$

$$\text{Heading} = -\frac{\psi}{\pi} * 180 + 90 \qquad (7.2)$$

The values of both psi and heading are cyclical in that once they attain the maximum value, they return to zero on the next step. This means one actual heading *H* can be represented by any number of values $H + 360*n$ where n can take any positive or negative integer value. The same can be said for the values of psi where any actual ψ_1 can also be represented by $\psi_1 + 2\pi n$ where n can take any positive or negative integer value. To eliminate confusion and mathematical discrepancies, the values for heading are limited to [0:360] by adding or subtracting 360, while the values for psi are limited to [0:2π] by adding or subtracting 2π as needed. The values for the desired heading and desired psi output from the Helm IvP are also limited in this manner.

It was found to be a problem when the error between desired psi and actual psi was greater than π or less than - π, as the controller would command the vehicle to rotate in the wrong direction. An example is shown in Fig. 15 where the desired psi is 1.9π and the actual psi is 0.1π, yielding an error of 1.8π with equation (7.3).

$$\psi\ Error = Desired\ Heading - Actual\ Heading \quad (7.3)$$

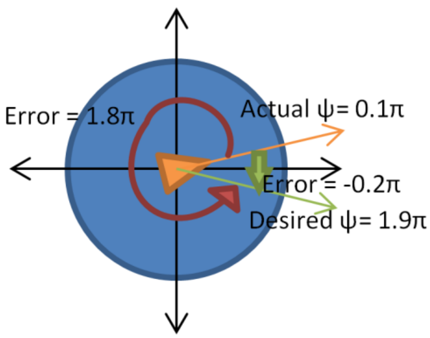

Fig. 15. Example of ψ error problem

In this example case the vehicle would be commanded to rotate to correct the 1.8π it thinks is error, when in fact the error is only -0.2π if it rotates in the other direction to correct. This can cause the system to become unstable and the vehicle can spin out of control. The main problem is when the system is trying to achieve a psi of zero because if it is overshot by even a small amount then the vehicle will see almost 2π of error again and will control the rotation speed up to compensate, causing zero to be overshot even more the next time. This repeats until the vehicle is at its maximum yaw rotation rate just spinning in place. This problem is resolved by limiting the psi error value to +/- π by adding or subtracting 2π when outside of these limits. In this manner, the vehicle will always correct in the direction with the least magnitude of error, resolving the problem.

VIII. PID Controller

The VideoRay UUV has internal low level controllers that allow the user to command the vehicle to a constant speed, depth, and heading. In order to simulate these three low level controllers, three PID controllers were used. A PID controller is a Proportional, Integral and Derivative controller, referring to the three components of error information used to calculate the control values. The proportional component of the controller uses the value of the error between desired and actual heading, speed or depth to proportionally set the control values. For example, if the error in depth was 10 meters and the proportional term constant was 2, the output of this part of the controller would be 20 newtons set to the desired Z force. This is intuitive because the larger the error is, the faster the thrusters should be controlled to spin.

The other two terms in the controllers are not as obvious, but yield great benefits when used correctly. The integral term is the sum of all previous error values over time. This term eliminates any steady state error from the system. An example of steady state error can be seen in the speed control of the UUV. When only proportional control is used for the speed control, the vehicle will settle to a speed that is around 10% of the desired speed, giving a steady state error of 90% of the desired value. This steady state error is eliminated with the addition of the integral term in the controller for desired X force. The reasoning behind the integral term is that any steady state error will cause the sum of the error terms to grow larger and larger until the integral term has a significant effect on the final output of the control variable. The PID controller for desired X force is the only one of the three used in this simulation that utilizes an integral component.

The final term in the PID controller is the derivative term, which is used by both the depth and the heading controllers in this system. The derivative term of the error represents the difference between the current error and the previous error. This term has the effect of damping the overshoot and settling time of the system response, without decreasing the rise time. This means that the system should get to the desired heading and depth very quickly without overshooting and having to overcorrect repeatedly until the desired value is obtained. A good example of what this looks like is shown in the vehicle speed response Fig. 16.

The vehicle response simulation described in this paper in the Matlab Simulation section yielded plots of the responses of these three PID controllers. Fig. 16 shows the speed response of the vehicle when commanded to a speed of one m/s at time t=10 seconds and then back to zero m/s at time t=100 seconds. This response is very desirable as it has a very quick rise time to the desired speed with very little overshoot or ringing. The spike at time t ~12 seconds results from the thrusters having to sacrifice speed for vehicle rotation as the vehicle is commanded to turn 180°. Fig. 17 shows the vehicle heading or psi response when commanded to π at time t=10 seconds and to π/4 at t=100 seconds. The rise time is sufficiently fast and there is a small amount of overshoot which is acceptable, and almost zero ringing. Fig. 18 shows the vehicle depth response when commanded to a depth of 10 meters. The slow rise time seen in this response stems from the fact that there is only one thruster available to control the vertical movement of the vehicle, limiting vertical speed.

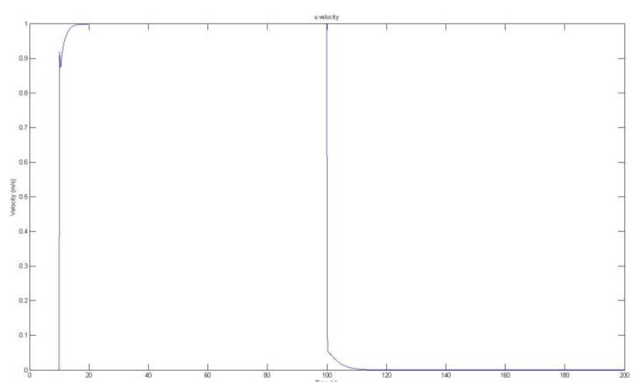

Fig. 16. Forward velocity response of the vehicle when commanded to a desired speed of 1 m/s at t=10s

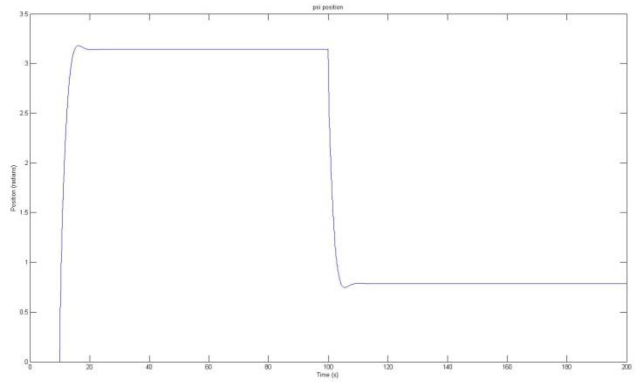

Fig. 17. Vehicle heading or ψ response when commanded to a desired ψ of π at t=10s and π/4 at t=100s

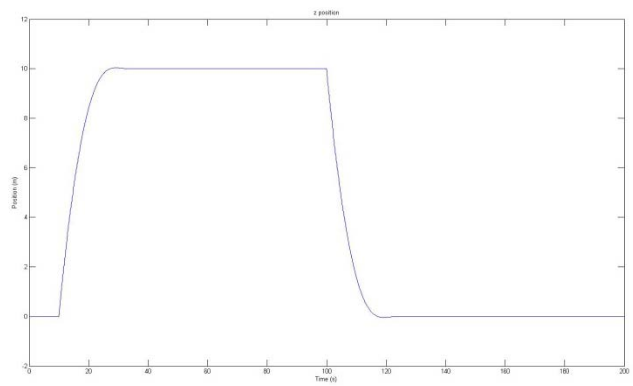

Fig. 18. Vehicle depth response when commanded to a desired depth of 10 meters at t=10s

IX. Conclusion and Future Work

This research presented the control, modeling and simulation of the unmanned underwater vehicle VideoRay Pro III. A Matlab simulation of this UUV graphically presented the response of the vehicle to desired depth, speed and heading commands. The hydrodynamic model of the vehicle was presented, along with the formulas relating the desired forces and torques applied to the vehicle with the control values for the three thrusters present on the VideoRay. The low level PID controllers as well as the guidance controller were both described in detail. The results of the Matlab simulation as well as the final system testing are presented in this paper. The system works well in testing and is able to simulate the vehicle successfully to run a mission utilizing all software modules developed for the vehicle.

The end goal for this project is to have an unmanned underwater vehicle that is able to autonomously track cables and discover other man-made objects on the sea floor. Future research on this project includes work on simulation and implementation of simultaneous localization and mapping (SLAM). Adding SLAM to the system would allow for vehicle position and speed determination in a state of sensor deprivation. While underwater, the vehicle is not able to determine its exact location using GPS and must rely on dead reckoning using data from a compass and inertial measurement unit. SLAM facilitates more accurate position determination as well as area mapping. Another topic of future work is to implement the MOOS IvP Helm control on the actual vehicle and analyze the real-world system performance to compare with the simulated response. A final piece of future work would be research on underwater image and sonar processing for use in cable tracking, position determination and guidance, man-made object detection, and other future mission capabilities.

Acknowledgment

The authors would like to thank Kevin DeMarco for his help and guidance through this research, as well as the Georgia Tech Research Institute and Georgia Institute of Technology for the resources provided in order to conduct this work.

References

[1] W. Wang, "Autonomous Control of a Differential Thrust Micro ROV," University of Waterloo, Mechanical Engineering Department, Waterloo, Canada, October 2006.

[2] T. Fossen, Guidance and Control of Ocean Vehicles, John Wiley and Sons: New York, 2nd edition, 1994, pp.5–56.

[3] M. Benjamin, H. Schmidt, P. Newman and J. Leonard, "An Overview of MOOS-IvP and a Users Guide to the IvP Helm," MIT, Department of Mechanical Engineering, Computer Science and AI Laboratory, Cambridge, MA, Department of Engineering Science, University of Oxford, England, May 2013.

[4] M. Benjamin, "MOOS-IvP Autonomy Tools Users Manual," Release 13.5, MIT, Department of Mechanical Engineering, Computer Science and AI Laboratory, Cambridge, MA, May 2013.

[5] T. Milat, "Simulator Plovila Temeljen Na Moos," University of Zagreb, Computer Science Department, Zagreb, Croatia, June 2010.

[6] J. Milgram, "Strip Theory for Underwater Vehicles in Water of Finite Depth," MIT, Department of Mechanical Engineering.

[7] VideoRay, "Pro 3 Product Information," VideoRay LLC.

[8] VideoRay, "VideoRay Pro 4 Operator Manual," Version 1.03.01, VideoRay LLC, 2010.

[9] ROS, www.ros.org.

[10] P. Newman, "MOOS – Mission Oriented Operating Suite," Version 2.1, MIT, Department of Ocean Engineering, Oxford University, Department of Engineering Science, March 2006.