Complete SE[superscript 3] underwater robot control with arbitrary thruster configurations

Citation: Doniec, Marek et al. "Complete SE[superscript 3] underwater robot control with arbitrary thruster configurations." in Proceedings of the 2010 IEEE International Conference on Robotics and Automation Anchorage Convention District, May 3-8, 2010, Anchorage, Alaska, USA IEEE, 2010. 5295-5301.

As Published: http://dx.doi.org/10.1109/ROBOT.2010.5509538

Publisher: Institute of Electrical and Electronics Engineers

Persistent URL: http://hdl.handle.net/1721.1/67312

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Complete SE^3 Underwater Robot Control with Arbitrary Thruster Configurations

Marek Doniec, Iuliu Vasilescu, Carrick Detweiler, Daniela Rus
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts
{doniec, iuliuv, carrick, rus}@csail.mit.edu

***Abstract*— We present a control algorithm for autonomous underwater robots with modular thruster configuration. The algorithm can handle arbitrary thruster configurations. It maintains the robot's desired attitude while solving for translational motion. The attitude can be arbitrarily chosen from the special orthogonal group SO^3 allowing the robot all possible orientations. The desired translational velocities can be chosen from R^3 allowing the robot to follow arbitrary trajectories underwater. If the robot is not fully holonomic then the controller chooses the closest possible solution using least squares and outputs the error vector.**

We verify the controller with experiments using our autonomous underwater robot AMOUR. We achieve roll errors of 1.0 degree (2.1 degrees standard deviation) and pitch errors of 1.5 degrees (1.8 degrees standard deviation). We also demonstrate experimentally that the controller can handle both non-holonomic and fully holonomic thruster configurations of the robot. In the later case we show how depth can be maintained while performing 360 degree rolls. Further, we demonstrate an input device that allows a user to control the robot's attitude while moving along a desired trajectory.

I. INTRODUCTION

We have developed a small, highly maneuverable, underwater autonomous vehicle called AMOUR [1]. AMOUR is designed to be a flexible underwater platform that can operate in shallow ocean environments. The robot's tasks include deploying static sensor nodes, taking pictures and videos of coral reefs for environmental monitoring, performing visual and acoustic ship-hull inspection, and docking with other underwater vehicles.

This wide range of tasks requires a modular robot whose thrusters can be easily repositioned, added, or removed from the system to ensure the system can deliver and position the payload with the precision required by the task. For example, adding an underwater imaging system, such as the one described in [2], significantly changes the system dynamics and the thrusters have to compensate. Using the robot to collect and deploy sensor nodes [3] also requires a robot configuration and control system that is very adaptive. Other tasks, such as optical data muling from deployed sensors [4] require a stable controller with high positioning precision.

The prior version of our robot required a different controller for each of these tasks [1]. The controller had a separate proportional-integral-derivative (PID) control loop for each of the pitch, roll, and yaw angles of the robot. This controller requires a knowledge of the contribution of each thruster or control surface to the pitch, yaw, and roll of the robot. However, we found it difficult to add, remove or move thrusters as the mission requirements changed. Additionally, using the typical approach of Euler angle rotation matrices yields controllers which are only locally stable and are subject to gimbal lock. For these reasons we wish to develop a new controller that can adapt to different numbers and orientations of the thrusters.

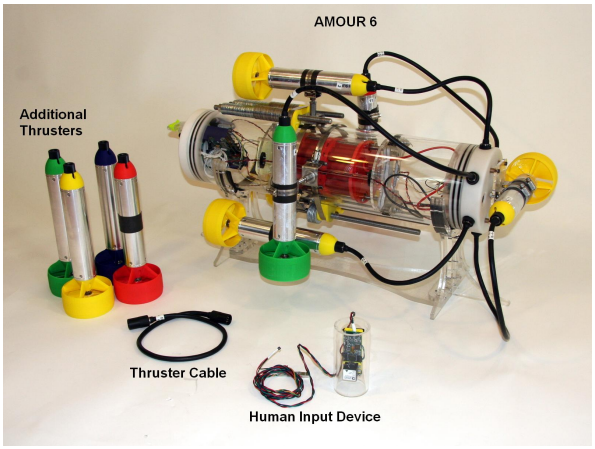


Fig. 1. The robot assembly with 10 thrusters and the human input device (cylindrical object in the front). Five thrusters are attached in the same configuration as in Figure 2 and a sixth thruster is attached to the front of the robot to make it fully holonomic. The other four thrusters are for replacement. The left side of the robot contains the IMU and the controllers. The battery is visible in the middle (red). The right side contains the power electronics. Below the top thruster and at the bottom of the robot the external ballast is visible that is necessary to weight down the robot. The robot is over 3kg buoyant without any payload attached.

In this paper we describe a new Modular Thruster Control Algorithm that supports an arbitrary number of thrusters and thruster configurations and can effect arbitrary orientations on a robot with a modular thruster system. The orientation of the robot can be arbitrarily set and changed while maintaining independent control of the translational components (e.g. the robot can be rolling head-over-heels while maintaining depth). More specifically, the attitude can be arbitrarily chosen from the special orthogonal group SO^3 (rotation group for three-dimensional space) while choosing the translational velocities from R^3.

We also describe a human input device that allows for very natural control of an underwater robot. The input device is held in the user's hand. The orientation of the robot follows that of the input device. The human can use the device to guide the robot's direction and orientation. For example, to survey a coral-head with the camera pointed at the coral, the user aims the input device to keep the nose/camera of the robot pointed at an area of interest. As the robot moves, the user sets its desired orientation and the controller effects the direction of travel along the desired vector.

This paper is organized as follows. We start with a survey of the related work. We then present the algorithm for modular thruster control. We discuss the implementation of the algorithm on the AMOUR underwater vehicle and present the results of experiments demonstrating the stability of our controller in many different orientations. Finally, we show that the controller supports the addition of another thruster to the robot.

II. RELATED WORK

Most underwater vehicles are torpedo shaped with only one or two thrusters. Traditional pitch/roll/yaw controllers work well on these types of configurations. Our robot, AMOUR, is a high degree of freedom robot [1] that can accommodate up to 8 thrusters. Other robots in this class of robots include the University of Hawaii ODIN-III robot [5], the CSIRO Starbug robot [6] and the Bluefin Robotics HAUV [7].

The controller described in this paper is most closely related to work by Hanai *et al.* [8]. They propose a similar geometric solution to robot control. The desired translational and rotational forces are used to determine the force for each thruster using a least squares method. This work assumes known thruster parameters and requires that the thrust vs. voltage curve to compute desired output values is known. Further, the controller does not have a derivative component leading to large oscillations and often needs more than 20 seconds to settle. Our system does not assume calibrated thrusters and adds derivative control to produce stable control loops.

Ghabcheloo *et al.* uses a similar vehicle orientation model to steer a group of robots to maintain formation. This work is done in simulation and does not address individual robots or thrusters [9]. Oh *et al.* describes how to compute paths for homing and docking using a similar vehicle orientation representation [10].

Fossen *et al.* present a survey of methods for control allocation of overactuated marine vessels [11]. The systems are not underactuated and this work focuses on the two dimensional case where depth, pitch, and roll are not considered. However the model could be extended to the three dimensional case.

Lee *et al.* uses Euler angles to create a 6-DOF controller for a simulated underwater vehicle. A genetic algorithm tunes the controllers [12]. This controller is subject to gimbal lock, however, their approach to tuning the individual controllers may be advantageous. Zhao *et al.* proposes an adaptive controller which does not require tuning by a human [13]. They emphasize the importance of being able to maintain a stable controller even when vehicle dynamics change.

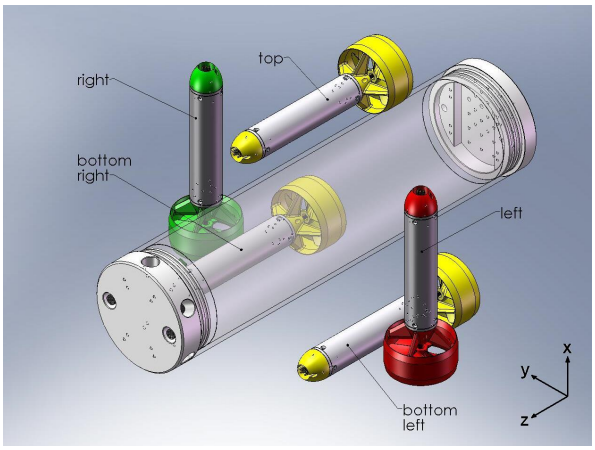


Fig. 2. AMOUR's standard configuration with 5 thrusters. The thrusters can be arbitrarily mounted along the body. The body measures 17.8cm in diameter and 72.4cm in length. The thruster measure 4.8cm in diameter and 25.4cm in length. They drive a 9.4cm diameter and 9.6cm pitch propeller which is housed in a 10.1cm diameter nozzle.

III. MODULAR THRUSTER CONTROL ALGORITHM

This section describes the Modular Thruster Control Algorithm. We first state the assumptions, the necessary input data, and what is provided as output. Then we give a high level explanation of how we achieve complete control in the special euclidean group SE^3 (group of all translations and rotations in three-dimensional space). Finally, we present the details of the Modular Thruster Control Algorithm.

A. Assumptions, input, and output

We assume that the robot knows the position and orientation of the thruster's and that the robot is capable of determining its own attitude. In particular, the algorithm presented takes as input a measured acceleration vector, $\mathbf{a} \in \mathcal{R}^3$, and a measured magnetic field vector, $\mathbf{m} \in \mathcal{R}^3$. We assume that the acceleration vector is pointing towards the ground and that the magnetic field vector is pointing north and upwards. Further, we assume the robot has a pressure sensor to measure the robot's depth, $d_{is} \in \mathcal{R}$. At sea level depth is assumed to be 0 meters and as the robot dives the value d_{is} decreases.

The control algorithm presented in this paper takes the current and desired robot attitude and depth as input and outputs speed commands for each thruster. The current state of the robot is computed from IMU measurements.

The goal attitude is provided as a rotation matrix and the goal depth is provided as a scalar:

$$\mathbf{R}_{goal} \in \mathcal{SO}^3, \quad d_{goal} \in \mathcal{R}. \tag{1}$$

The algorithm is configured by providing a list of thruster positions, $\mathbf{p}_i \in \mathcal{R}^3$, and orientations, $\mathbf{o}_i \in \mathcal{R}^3$, in the robot coordinate frame for the N thrusters. Further, the

proportional, integral, and derivative constants for the PID controller need to be provided. Adding or removing thrusters is as simple as updating the position and orientation vectors and specifying the PID loop.

The center of mass is assumed to be at the origin of the robot coordinate system in this paper. However, the algorithm can easily be extended to incorporate a moving center of mass by simply subtracting the center of mass at the appropriate places.

The output of the main stage of the algorithm (the parts described in detail in this section) is an error for each thruster that is derived from the robot's rotational error and an error for each thruster that is derived from the robot's translational error.

B. Overview

The Modular Thruster Control Algorithm computes the rotational error in radians between the robot's current pose and desired pose. It also computes the axis around which this rotation should occur. Using a unit torque representation for all thrusters we solve for individual thruster errors that when summed will result in the total current rotational error.

Next, we compute the translational error given the thruster outputs using depth and position.[1] Using unit direction vectors for each thruster we solve for individual thruster errors that when multiplied with each thruster's direction will result in the total translational error.

The computed rotational and translational errors are each used as input to a separate PID controller on each thruster. Thus we can tune a thruster's response to translational and rotation errors independently. Finally, the output of both PID controllers is summed for each thruster to generate the thruster output.

To visualize why a separate rotational and translational PID controller is necessary for every thruster, consider the robot assembly shown in Figure 2. We remove the top and two bottom thrusters leaving only the left and right thruster and then try to roll the robot along its axis. We will encounter very little resistance as essentially no displacement of water occurs as the robot's body rolls. Thus we will only require a small P-gain for rotation control of the left and right thrusters. However, if we try to change depth with this two thruster configuration then we have to move the entire robot body broadside through the water. This will create a lot of drag and will require a far higher P-gain for the translational control of the two thrusters.

While we do not directly consider robot dynamics the use of two separate PID controllers for each thruster allows for a good approximation of simple robot dynamics. For example, in the above case we can tune the PID controllers derivative parameter to account for the momentum that occurs when the robot turns to prevent overshoot.

Our method is related to, but differs from Hanai *et al.* [8] in that they directly compute the required thrust necessary to be produced by each thruster. Hanai *et al.* assume that the voltage-to-thrust curve is known and drive each thruster directly from this curve. However, such a curve is static and will not accommodate changes in the vehicle's dynamics. Our method allows the user to quickly adjust the controller to such changes.

[1]In the experiments presented in this paper we did not have robot XY-position and so computed our translational error only based on depth.

C. Computing the current robot state

Using the data provided by the IMU we first compute the robot's attitude by computing the east, north, and up vectors in its frame of reference and then generate the respective rotation matrix:

$$\begin{gathered}\mathbf{e} = \mathbf{a} \times \mathbf{m}, \quad \mathbf{n} = \mathbf{e} \times \mathbf{a} \\ \mathbf{R}_{is} = [\hat{\mathbf{e}}, \hat{\mathbf{n}}, -\hat{\mathbf{a}}]\end{gathered} \tag{2}$$

D. Computing the attitude error

Next we compute the rotation that will take the robot from its current attitude to the goal attitude ($\mathbf{R}_{is} \rightharpoonup \mathbf{R}_{goal}$). This rotation has to be represented in the robot's local coordinate frame. It turns out that the command will be:

$$\mathbf{R}_{cmd} = (\mathbf{R}_{goal}^{-1} \cdot \mathbf{R}_{is})^{-1} \tag{3}$$

We then convert this into a desired rotation angle and axis:

$$\omega = \arccos((\mathrm{Trace}(\mathbf{R}_{cmd} - 1)/2) \tag{4}$$

$$\mathbf{r} = \frac{1}{2 \cdot \sin(\omega)} \cdot \begin{bmatrix} \mathbf{R}_{cmd,32} - \mathbf{R}_{cmd,23} \\ \mathbf{R}_{cmd,13} - \mathbf{R}_{cmd,31} \\ \mathbf{R}_{cmd,21} - \mathbf{R}_{cmd,12} \end{bmatrix} \tag{5}$$

where $\mathbf{R}_{cmd,xy}$ represents entry (x, y) of $\mathbf{R}_{cmd}$.

Next we compute the depth error in meters and the torque error vector. The direction of the torque error vector represents the torque axis of the error, while the length gives the amplitude of the error in radians:

$$e_{robot,depth} = d_{goal} - d_{is} \tag{6}$$

$$\mathbf{e}_{robot,rot} = \omega \cdot \mathbf{r} \tag{7}$$

E. Computing separate thruster errors for rotation

To solve for rotation we compute the torque vector for every thruster:

$$\mathbf{t}_i = \mathbf{p}_i \times \mathbf{o}_i, \quad i \in 1 \ldots N. \tag{8}$$

We then concatenate all torque vectors to form an equation system. Extra user constraints, like the definition of thruster symmetries, can be added as extra lines $\mathbf{C}_{rot}$ to the equation system:

$$\mathbf{A}_{rot} = \begin{bmatrix} \mathbf{t}_1 & \ldots & \mathbf{t}_N \\ & \mathbf{C}_{rot} & \end{bmatrix} \tag{9}$$

To finally compute the error for every thruster we use linear least squares approximation. The advantage of this method is that if a perfect solution does not exist, for instance if the robot does not have enough degrees of freedom (DOF),

then we get the best possible solution. To compute the result we first compute the Moore-Penrose pseudoinverse:

$$\mathbf{A}_{rot}^{+} = \mathbf{A}_{rot}^{T} \cdot (\mathbf{A}_{rot} \cdot \mathbf{A}_{rot}^{T})^{-1} \tag{10}$$

and use it to solve for thruster errors. Because we added constrains $\mathbf{C}_{rot}$ to $\mathbf{A}_{rot}$ we need to pad $\mathbf{e}_{robot,rot}$ with zeros equal to the number of rows in $\mathbf{C}_{rot}$:

$$\mathbf{E}_{thrusters,rot} = \mathbf{A}_{rot}^{+} \cdot \begin{bmatrix} \mathbf{e}_{robot,rot} \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{11}$$

Note that because of thruster redundancy it is likely that infinitely many solutions exist. Adding constraints to $\mathbf{A}_{rot}$ are one way of choosing the right solution. Alternatively, as in [8] a specific solution can be chosen (for example to minimize the thruster output while avoiding thruster dead bands). We have chosen the constraint adding solution because in our experiences thruster dead bands did not affect the performance of our robot.

F. Computing separate thruster errors for translation

We only use the depth error $e_{robot,depth}$ to computer the translational error $\mathbf{e}_{robot,trans}$. However, provided position information for the robot an error in the XY plane can be computed and added to $\mathbf{e}_{robot,trans}$.

$$\mathbf{e}_{robot,trans} = [0 \quad 0 \quad e_{robot,depth}]^{T} \tag{12}$$

Using the concatenation idea that allowed us to create a system of equations from the torque vectors to solve for rotation, we can also concatenate all unit orientation vectors:

$$\mathbf{A}_{trans} = \begin{bmatrix} \mathbf{o}_1 & \dots & \mathbf{o}_N \\ \multicolumn{3}{c}{\mathbf{C}_{trans}} \end{bmatrix} \tag{13}$$

Again we use least square approximation to solve for thruster errors and remember to pad $\mathbf{e}_{robot,trans}$ with zeros equal to the number of rows in $\mathbf{C}_{trans}$:

$$\mathbf{A}_{trans}^{+} = \mathbf{A}_{trans}^{T} \cdot (\mathbf{A}_{trans} \cdot \mathbf{A}_{trans}^{T})^{-1} \tag{14}$$

$$\mathbf{E}_{thrusters,trans} = \mathbf{A}_{trans}^{+} \cdot \begin{bmatrix} \mathbf{e}_{robot,trans} \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{15}$$

G. Final steps

We described how we can compute a rotational and a translational error for every thruster. Each of these errors is used as input to a PID controller and the output of these controllers is added to produce a thruster output.

It is worth mentioning that thruster failures can be handled by removing the appropriate line from $\mathbf{A}_{rot}$ and $\mathbf{A}_{trans}$. Failure can be detected in several ways, for example the thruster not responding to communication requests. We also envision a system that actively can detect changes in thruster response by using a disturbance observer (DOB) similar to [13].

Further note that unless the number or position of thrusters changes (or if the constraints change) then $\mathbf{A}_{rot}$ and $\mathbf{A}_{trans}$ are constant and do not need to be computed during every update step.

A summary of the Modular Thruster Control Algorithm is given above in Algorithm 1.

Algorithm 1 Modular Thruster Control Algorithm

$\mathbf{A}_{rot}^{+} = \mathbf{A}_{rot}^{T} \cdot (\mathbf{A}_{rot} \cdot \mathbf{A}_{rot}^{T})^{-1}$
$\mathbf{A}_{trans}^{+} = \mathbf{A}_{trans}^{T} \cdot (\mathbf{A}_{trans} \cdot \mathbf{A}_{trans}^{T})^{-1}$
loop
  $\mathbf{R}_{is} = [\hat{\mathbf{e}}, \hat{\mathbf{n}}, -\hat{\mathbf{a}}]$
  $\mathbf{R}_{cmd} = (\mathbf{R}_{goal}^{-1} \cdot \mathbf{R}_{is})^{-1}$
  $\mathbf{r} = axis(R_{cmd}), \quad w = angle(R_{cmd})$
  $\mathbf{E}_{thr.,rot} = \mathbf{A}_{rot}^{+} \cdot [\omega \cdot \mathbf{r}^{T} \quad 0 \dots 0]^{T}$
  $\mathbf{E}_{thr.,trans} = \mathbf{A}_{trans}^{+} \cdot [0 \quad 0 \quad (d_{goal} - d_{is}) \quad 0 \dots 0]^{T}$
end loop

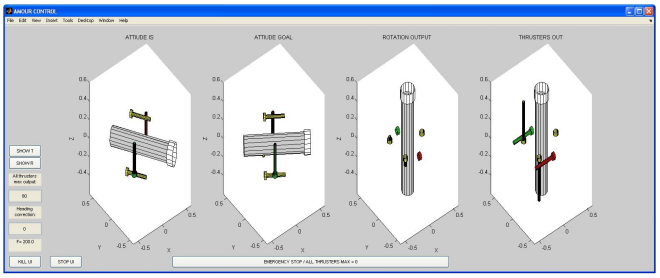


Fig. 3. AMOUR's user interface. The UI can run on a laptop connected to the robot through a serial cable. The UI provides visual feedback and allows for easy on-the-fly tuning of parameters.

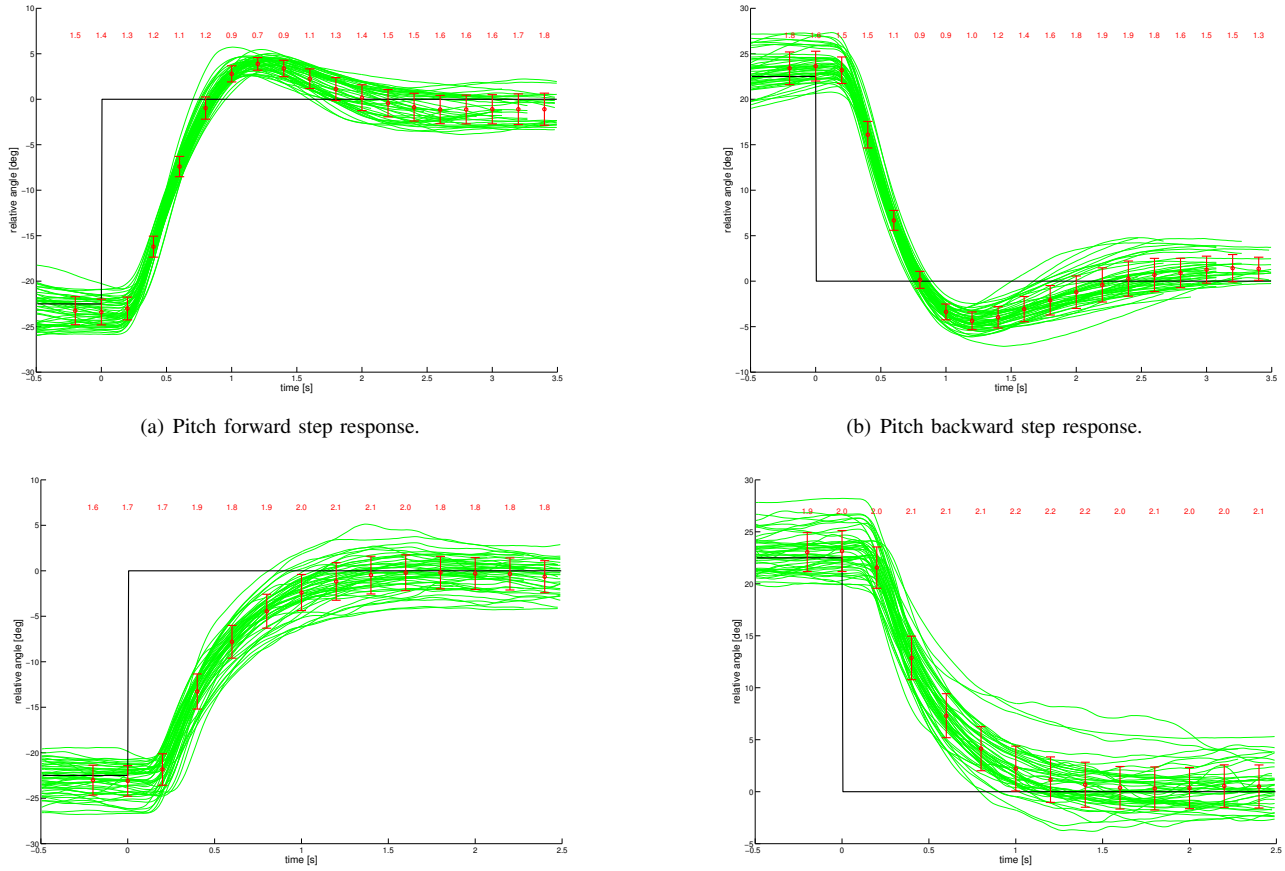
IV. HARDWARE

A. AMOUR 6

We implemented the Modular Thruster Control Algorithm on the sixth iteration of AMOUR (Autonomous Modular Optical Underwater Robot), our in-house developed AUV. The previous versions of AMOUR were presented in [1], [2], [14]. The main improvement in the current version is increased modularity.

AMOUR is composed of a main body and up to 8 external thrusters. Without payload and equipped with 5 thrusters the robot weights approximately 17.5kg and is 3kg buoyant.

The body is an acrylic cylinder, measuring 17.8cm in diameter and 72.4cm in length. The body houses a Lithium-Ion battery, battery management board, inertial measurement unit (IMU), sensor board [3], communication hub for serial devices, and a small PC [15]. The battery has a capacity of 645Wh and amounts for a third of the weight of the robot. The battery is actuated by an electric motor and it can travel along the axis of the robot to shift the robot's center of mass [1]. At each end, the body has a two 10cm empty section that can be used for additional dry payloads. The battery management board distributes the power, measures



(a) Pitch forward step response.

(b) Pitch backward step response.

(c) Roll counter-clockwise step response.

(d) Roll clockwise step response.

Fig. 4. Experimental results for pitch and roll step response. The robot was commanded to performed 48 discrete 22.5 degree turns (three full rotations) around both the pitch and roll axes. The black lines represent the step command and the red lines represent AMOUR's response. In the pitch direction the robot settles within 1.5 degrees (1.8 degree standard deviation) of the commanded angle in an average of 1.9 seconds. In the roll direction the robot settles within 1 degree (2.0 degree standard deviation) of the commanded angle in an average of 1.4 seconds.

the robot's current consumption and tracks the precise state of the battery using a charge counter. The IMU estimates the pose and the depth of the robot by fusing the raw data from 10 sensors (one pressure sensor, 3 magnetic field sensors, 3 accelerometers and 3 gyroscopes). The sensor board is used for (1) reading of GPS and marine analog and digital sensors, (2) data logging and (3) radio communications. The communication hub allows the connection of up to 16 devices (serial TLL, RS232 or RS485). The PC will be used in the future for control, high level mission planning, and online data processing of the sensory payload.

The thrusters are designed in house. Each thruster is composed of a motor controller and a 600W geared brushless DC motor driving a Kort nozzle propeller with a diameter of 9.4cm. The motor and the electronics are housed in an aluminum cylinder, with a diameter of 4.8cm and a length of 25.4cm. Each thruster can generate up to 4kg of static thrust. The thrusters receive commands via a RS485 bus.

AMOUR's configuration is very modular. The thrusters can be attached along the robot's body or to the robot's end caps. Each thruster is connected electrically to one of the 8 ports fitted on AMOUR's bottom cap. Replacing a thruster or changing the thruster configuration can be done in less than 5 minutes. AMOUR is designed to carry internal and external payloads. AMOUR's control algorithms and high power thrusters allow precise and fast manipulation of payloads of size similar to its own.

For the purpose of this paper, the robot was connected to a laptop computer on which we ran the control algorithm and the user interface shown in Figure 3. We used a full duplex RS232 link to relay the IMU pose estimation from the robot to the laptop and the thruster commands from the laptop to the robot. In the future we will implement the control algorithm on the IMU's processor.

B. Human Input Device

We built a miniature model of AMOUR fitted with an IMU, to be used as a human input device (Figure 1). The operator can use this device to command the robot's pose quickly and intuitively. When in use, the IMU inside the model reports its orientation, which can be transmitted to AMOUR's controller at a rate of 200Hz. We used this device to evaluate the robot's response to complex motions. In the future we envision researchers using this device to remote control the robot for video shooting, manipulation, and data gathering.

V. EXPERIMENTS

We evaluated the performance of the Modular Thruster Control Algorithm during a set of trials performed at the MIT Alumni swimming pool. The pool depth varies between 2 and 3 meters. We typically operated the robot at a depth of 1 meter. For the first set of experiments the robot was fitted with 5 thrusters as shown in Figure 2. At one point during the experiment one thruster failed. We were able to replace the thruster and resume operations in under five minutes. In the second set of experiments we added a sixth thruster as shown in Figure 1. We trimmed the buoyancy and the balance of the robot by adding stainless steel washers to 4 rods distributed around the robot's body. After trimming the robot was 50g negatively buoyant and almost neutrally balanced.

We then tuned the translation and rotation PID controllers for each of the five thrusters. As described in Section III, we need two different controllers for the translation and rotation to account for the fact that, for example, in the roll case there is very little drag, however, there is a lot of drag during depth translation.

We were able to take advantage of the symmetries of the robot to reduce the actual number of PID loops that we needed to tune to four (the two vertical and three horizontal thrusters have the same parameters). First, we stabilized the rotational controllers and then we tuned the translation controller. In our setup we did not have an XY-positioning system, so our translation controller just dealt with depth control. When we added a sixth thruster we were able to keep all of the parameters for the other thrusters the same and just tuned the parameters for the new thruster.

Our general approach to tuning the PID controllers was to increase the proportional term until they responded quickly and had small overshoots and oscillations. Then we added a derivative component to dampen the oscillations. Finally, an integral component can be added to compensate for any constant offsets, however, we did not find this necessary in these trials as the robot was well balanced. Tuning all of the parameters in this experiment took under 15 minutes and only needed to be done once.

Retuning individual thruster PID controllers is not necessary unless the position of the thrusters changes significantly, which never occured during our experiments. In our experience rotating the thruster by as much as 10 degrees did not visibly affect controller performance. It should also be noted that the thrusters were attached by hand and their positions were only estimated as opposed to measured. This shows the robustness of our controller to inaccuracies.

A. Experimental Results

We performed a number of experiments to validate the control algorithm. In the first experiment we examined the step response of the controller to changes in orientation. The robot was configured with five thrusters. For both clockwise and counterclockwise pitch and roll we commanded 48 discrete 22.5 degree turns (three full turns). The results of these experiments are shown in Figure 4. In the pitch direction the robot settles within 1.5 degrees of the commanded angle in an average of 1.9 seconds. In the roll direction the robot settles to within 1 degree in an average of 1.4 seconds. While we do not have room to report on the yaw controller, other experiments using the yaw controller yield similar results.

The change in the commanded angle is illustrated by the black square signal in the figure. Note that there is a 160 millisecond delay from the command changing to the robot responding. We suspect that this delay is caused by our user interface. The pitch controller initially overshoots but then quickly settles to the target angle. The roll controller does not overshoot. Impulses given to the robot by a human swimmer resulted in similar quick settling times.

Figure 5(a) shows the depth of the robot as the robot rolled with five thrusters. In this configuration the robot can only control five degrees of freedom, the sixth is uncontrolled. When the robot is on its side it does not have any thrusters to compensate for the slight negative buoyancy. Thus, the robot starts to sink in that orientation, as seen in Figure 5(a). In this situation the controller alerts the user that the desired command cannot be executed precisely and outputs the type and magnitude of the error.

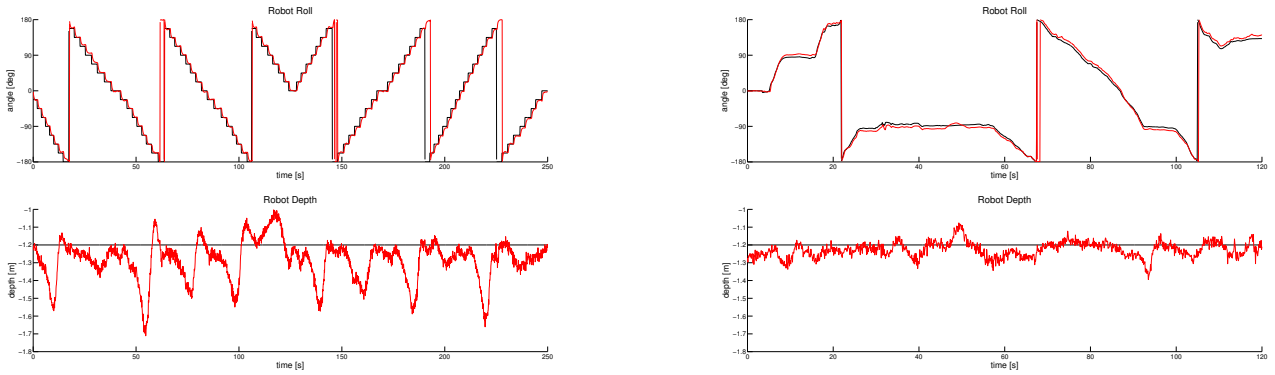
To make the robot approximately holonomic we added a sixth thruster to the nose of the robot in a horizontal configuration as shown in Figure 1. The robot is then able to maintain depth when it is rotated on its side. However, in doing so the bow thruster exerts a torque on the robot that the control algorithm compensates for by using the other thrusters. Figure 5(b) shows the depth of the robot as the robot rolls with six thrusters. This plot shows that we are able to maintain depth control of the robot even when it is on its side. In this experiment the robot was controlled using the human input device, keeping the robot on its side most of the time where it is most difficult to maintain depth.

The ability of the robot to track the desired trajectory of the human input device can be seen in the top portion of Figure 5(b). The black line indicates the target orientation based on the input device and the red line is the actual orientation of the robot. Most of the time these lines are completely overlapping indicating that the robot achieves the desired orientation. The video attached to this paper shows the use of the input device. Multiple users operated the robot in the pool environment with the input device. All users reported that robot could be easily and intuitively controlled. Complex rolls and orientations can easily be achieved due to the high update rate of the input device and the quick response time of the robot.

VI. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this paper we described the Modular Thruster Control Algorithm for autonomous underwater robots with modular thruster configuration. This algorithm can orient a robot in any configuration in SE^3. We describe how the control algorithm maps a desired change in robot state to individual thruster outputs. The algorithm accommodates the addition, removal, and arbitrary positioning of thrusters.



(a) Depth control at various roll angles, using 5 thrusters.

(b) Depth control at various roll angles, using 6 thrusters.

Fig. 5. This graph shows the performance of the control algorithm in two configurations: (a) 5 thrusters and (b) 6 thrusters. The top graphs show the roll angle and the bottom graphs show the depth. The black lines plot the roll and depth commands and the red lines plot the actual roll and depth. When using 5 thrusters, the robot is not fully holonomic and cannot control depth when rolled 90 degrees. At this angle the robot sinks due to its negative buoyancy. When a sixth thruster is attached the robot become approximately holonomic. In this configuration the robot can control its depth in any orientation.

We show experimentally that our robot quickly achieves the target configurations with little or no oscillations. We are able to maintain accurate depth control even when performing barrel or head-over-heels rolls. Making use of this control algorithm on our robot gives us a robust and stable platform that can be reconfigured for a variety of tasks including deploying sensor nodes, taking pictures and videos, harbor and port security, and docking with other vehicles.

We also presented a human input device which enables quick and intuitive control of an underwater vehicle. This input device allows a user to arbitrarily orient the robot even while it is moving along a motion vector.

B. Future Work

We plan to extend our control algorithm to accommodate the docking of multiple robots together to form one larger robot. By simply sharing thruster position information we should be able to control the new, larger, vehicle using the same algorithm.

One limitation of our algorithm is that it still requires parameter tuning for the PID values. While this is relatively quick to perform, we also plan to explore algorithms to automatically tune these parameters. Taking this a step further, we hope to learn the position and parameters for new thrusters which are added to the robot or automatically compensate for a failed thruster. We plan to allow for both self-tuning of the thrusters as well as adaptive tuning in case the position or parameters for a thruster change unexpectedly during operation.

Finally, we plan to extend the functionality of our human input device to allow for translational motion control.

VII. ACKNOWLEDGMENTS

We are grateful to DSTA Singapore and Intel for supporting in part this research.

REFERENCES

[1] I. Vasilescu, C. Detweiler, M. Doniec, D. Gurdan, S. Sosnowski, J. Stumpf, and D. Rus, "Amour v: A hovering energy efficient underwater robot capable of dynamic payloads," *International Journal of Robotics Research (IJRR)*, 2010.

[2] I. Vasilescu, "Using light underwater: Devices, algorithms and systems for maritime persistent surveillance," Ph.D. dissertation, MIT, February 2009. [Online]. Available: http://iuliu.com/pub/iuliu-vasilescu-phd-eecs-2009.pdf

[3] C. Detweiler, I. Vasilescu, and D. Rus, "An underwater sensor network with dual communications, sensing, and mobility," *OCEANS 2007 - Europe*, pp. 1–6, June 2007.

[4] M. Dunbabin, P. Corke, I. Vasilescu, and D. Rus, "Data muling over underwater wireless sensor networks using an autonomous underwater vehicle," in *Proc. IEEE ICRA 2006*, Orlando, Florida, May 2006, pp. 2091–2098.

[5] H. Choi, A. Hanai, S. Choi, and J. Yuh, "Development of an underwater robot, ODIN-III," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, 2003, pp. 836–841 vol.1.

[6] M. Dunbabin, J. Roberts, K. Usher, G. Winstanley, and P. Corke, "A hybrid AUV design for shallow water reef navigation," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 2105–2110.

[7] J. Vaganay, M. Elkins, D. Esposito, W. O'Halloran, F. Hover, and M. Kokko, "Ship hull inspection with the HAUV: US navy and NATO demonstrations results," in *OCEANS 2006*, 2006, pp. 1–6.

[8] A. Hanai, H. T. Choi, S. K. Choi, and J. Yuh, "Experimental study on fine motion control of underwater robots," *Advanced robotics: the international journal of the Robotics Society of Japan*, vol. 18, no. 10, pp. 963–978, 2004.

[9] R. Ghabcheloo, A. P. Aguiar, A. Pascoal, C. Silvestre, I. Kaminer, and J. Hespanha, "Coordinated path-following in the presence of communication losses and time delays," *SIAM - Journal on Control and Optimization*, vol. 48, no. 1, pp. 234–265, 2009.

[10] K. Oh, J. Kim, I. Park, J. Lee, and J. Oh, "A study on the control of AUV's homing and docking," in *9th IEEE Conference on Mechatrinics and Machine Vision in Practice*, ser. 9, Thailand, 2002, pp. 45–52.

[11] T. Fossen and T. Johansen, "A survey of control allocation methods for ships and underwater vehicles," in *Proceedings of the 14th IEEE Mediterranean Conference on Control and Automation*, Ancona, Italy, June 2006.

[12] J. Lee, M. Roh, J. Lee, and D. Lee, "Clonal selection algorithms for 6-DOF PID control of autonomous underwater vehicles," in *Artificial Immune Systems*, 2007, pp. 182–190.

[13] S. Zhao, J. Yuh, and H. Choi, "Adaptive DOB control of underwater robotic vehicles," in *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, vol. 1, 2001, pp. 397–402 vol.1.

[14] I. Vasilescu, P. Varhavskaya, K. Kotay, and D. Rus, "Autonomous modular optical underwater robot (amour) design, prototype and feasibility study," in *Proc. IEEE ICRA 2005)*, Barcelona, Spain, 2005, pp. 1603–1609.

[15] CompuLab. (2009) fit-pc2 wiki. [Online]. Available: http://fit-pc2.com/