Estimation of Thruster Configurations for Reconfigurable Modular Underwater Robots

Marek Doniec, Carrick Detweiler, Daniela Rus

Abstract We present an algorithm for estimating thruster configurations of underwater vehicles with reconfigurable thrusters. The algorithm estimates each thruster's effect on the vehicle's attitude and position. The estimated parameters are used to maintain the robot's attitude and position.
The algorithm operates by measuring impulse response of individual thrusters and thruster combinations. Statistical metrics are used to select data samples. Finally, we compute a Moore-Penrose pseudoinverse, which is used to project the desired attitude and position changes onto the thrusters.
We verify our algorithm experimentally using our robot AMOUR. The robot consists of a main body with a variable number of thrusters that can be mounted at arbitrary locations. It utilizes an IMU and a pressure sensor to continuously compute its attitude and depth. We use the algorithm to estimate different thruster configurations and show that the estimated parameters successfully control the robot. The gathering of samples together with the estimation computation takes approximately 40 seconds. Further, we show that the performance of the estimated controller matches the performance of a manually tuned controller. We also demonstrate that the estimation algorithm can adapt the controller to unexpected changes in thruster positions. The estimated controller greatly improves the stability and maneuverability of the robot when compared to the manually tuned controller.

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

1 Introduction

We wish to develop underwater robot systems whose dynamic properties can be adjusted in real time. Specifically, we are developing modular underwater robots capable of collecting and placing loads, and whose thruster configuration can be adjusted on the fly, by adding, removing, or repositioning thrusters. A key challenge is to develop robot control systems capable of adapting in situ, without human reprogramming to changing robot dynamics. In this paper we present an algorithm for estimating the thruster configuration of a modular underwater robot.

This line of research is motivated by our prior work on developing modular underwater robots with variable thruster configurations and dynamics. AMOUR is designed to be a flexible underwater platform that can operate in shallow ocean environments [11]. Other hovering underwater robots that maneuver predominantly using thrusters include the University of Hawaii ODIN-III robot [1], the CSIRO Starbug robot [3] and the Bluefin Robotics HAUV [10]. The Modular Thruster Control Algorithm developed in our previous work [2] provides attitude and position control for modular robots and can handle arbitrary thruster configurations. However, the algorithm requires knowledge of the thrusters' locations. The estimation algorithm in this paper determines the robot's inverse thruster model automatically effectively learning the robot's thruster geometry and dynamics.

Prior work on learning the thruster configuration of a robot uses neural nets to create a mapping between commands and thruster outputs. Van de Ven *et al.* give an overview of neural network control of underwater vehicles and simulation studies [13]. Gua *et al.* and Ishii *et al.* learn yaw control [6, 7] and Farrell *et al.* and Kodogiannis *et al.* learn depth control [4, 8]. They all verify their controllers in simulation as well as through experiments. Further, Lorentz *et al.* present an online adaptive single input single output neural net controller for the heave motion of the ODIN underwater vehicle [9]. They experimentally verify their controller in the pool. Gaskett *et al.* present a model-free reinforcement learning system for their underwater vehicle Kambara[5]. They evaluate their model in simulation with a two-thruster AUV (one left, one right) in the plane.

In this prior work, the neural nets attempt to simultaneously learn a model of the vehicle and a controller. The learning often takes a long time and the resulting controllers show slow response times of multiple seconds. In contrast to these works we assume the existence of a stable control algorithm and learn an inverse model of the thrusters that is then used by the controller. The approach is related to a procedure presented by Van de Ven *et al.* that uses neural nets to identify the main hydrodynamic parameters of an underwater vehicle model, assuming a priori knowledge of an inverse model of the vehicles propulsion system [12]. However, instead we determine the inverse thruster model and we do so analytically without the use of a neural net.

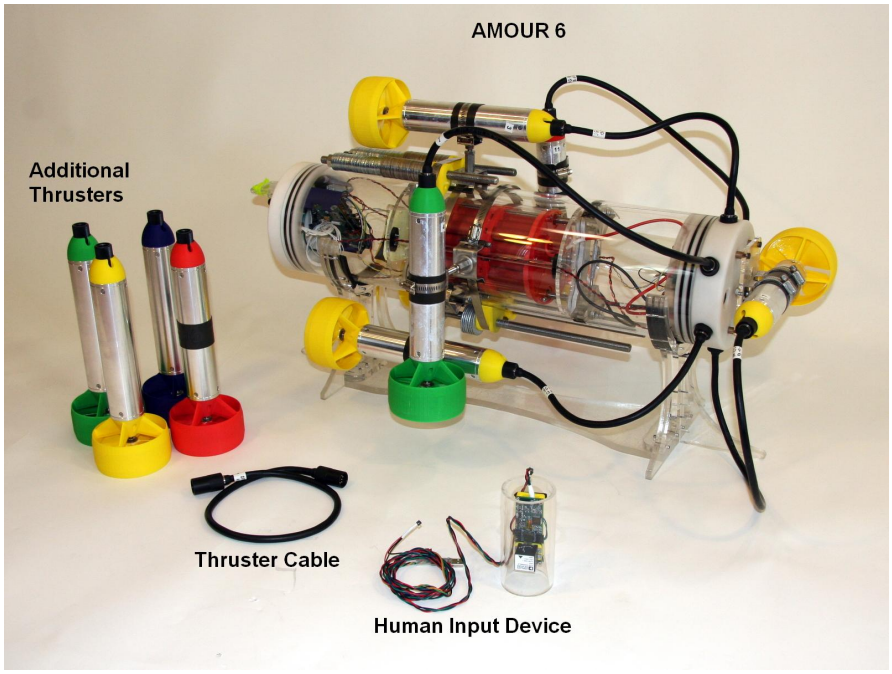


Fig. 1 Picture of the robot AMOUR 6 with spare thrusters.

2 Technical Approach

Algorithm 1 for estimating the thruster configuration of an underwater modular robot makes the following assumptions: (1) the robot is capable of determining its own attitude by measuring the acceleration vector, $\mathbf{a} \in \mathscr{R}^3$, and the magnetic field vector, $\mathbf{m} \in \mathscr{R}^3$; (2) the number of thrusters N is known but their location is not known; (3) each thruster is rigidly mounted to the robot at some unknown position with an unknown orientation of thrust; (4) thrusters take a command between -100 and $+100$ (full thrust backward/full thrust forward), 0 means the thruster is off. The output of Algorithm 1 is an inverse model of the thrusters, i.e. a mapping from a desired rotation and position change of the robot to thruster commands.

While the robot is at rest, the algorithm measures σ_{acc} and σ_{rot}, the acceleration noise levels and the changes in rotation of the robot. Estimation is performed by pulsing individual thrusters and recording the measured rotational changes *rot* and accelerations *acc* of the robot together with the thruster output. Each thruster is pulsed for 1 second with a settling time of 3 seconds in between pulses.

Data samples are only chosen during pulse time. Samples are selected when they show a change of at least k times larger than the noise level of the corresponding signal (σ_{acc} and σ_{rot}). In our implementations we use $k = 3$. Initially a thrust level of 50% maximum thrust is selected as the pulse strength. However, the flags $flag_{rot}$

and $flag_{acc}$ ensure, that if no samples are gathered during a pulse because the accelerations and changes to rotation are smaller than k times the noise levels, then the thrust level is increased by 10%.

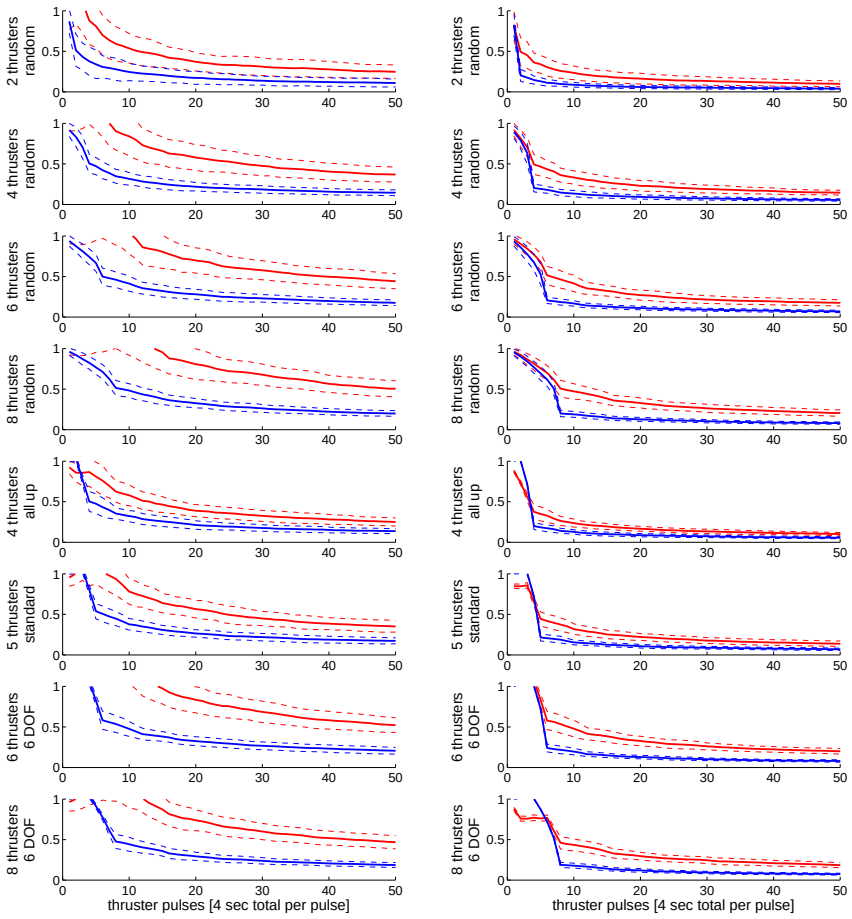
After a sufficient number of pulses (currently determined manually by an observer) we compute a least squares solution to the linear equation system $\mathbf{S} \cdot \mathbf{A} = \mathbf{T}$ for both accelerations and rotational changes. This is done by computing the Moore-Penrose pseudoinverse for the matrix $\mathbf{S}$. The matrices $\mathbf{A}_{acc}$ and $\mathbf{A}_{rot}$ represent inverse thruster models of the underwater vehicle. These matrices can be used to project the desired changes in attitude and position onto the robots thrusters. We refer interested readers to our previous work on the Modular Thruster Control Algorithm for details [2].

3 Simulations

We studied the thruster configuration estimation algorithm in simulation to verify the correctness of the estimation and its performance. We developed a model of the robot and considered configurations with 2, 4, 6, and 8 thrusters. In each case we selected some fixed thruster configurations to experiment with. We also ran the thruster estimation algorithm in segments of 100 runs for each thruster number (2, 4, 6, 8). For each of the 100 runs we generated randomly the thruster configuration. We

Algorithm 1 Thruster Configuration Estimation Algorithm

$thrust[1 \ldots N] \Leftarrow 0, \quad power \Leftarrow 0.5$
measure IMU noise levels σ_{acc} and σ_{rot}
while learning **do**
  PICK $I \in 2^N$
  $flag_{acc} \Leftarrow 1$, $flag_{rot} \Leftarrow 1$
  for 1 sec **do**
    $thrust[I] \Leftarrow power$
    if $||acc|| > k \cdot \sigma_{acc}$ **then**
      APPEND $(acc, thrust)$ TO $(\mathbf{S}_{acc}, \mathbf{T}_{acc})$
      $flag_{acc} \Leftarrow 0$
    end if
    if $||rot|| > k \cdot \sigma_{rot}$ **then**
      APPEND $(rot, thrust)$ TO $(\mathbf{S}_{rot}, \mathbf{T}_{rot})$
      $flag_{rot} \Leftarrow 0$
    end if
  end for
  if $flag_{acc}$ **or** $flag_{rot}$ **then**
    $power \Leftarrow \max(power + 0.1, 1.0)$
  end if
  $thrust[i] \Leftarrow 0, \quad$ PAUSE(3 sec)
end while
$\mathbf{A}_{rot} = \mathbf{S}_{rot}^{+} \cdot \mathbf{T}_{rot}, \quad \mathbf{A}_{trans} = \mathbf{S}_{acc}^{+} \cdot \mathbf{T}_{acc}$



(a) pulses at 20% power ($p = 20$) (b) pulses at 50% power ($p = 50$)

Fig. 2 Thruster estimation algorithm simulation results for 4 randomly drawn thruster configurations (4 top rows) and 4 manually defined thruster configurations (4 bottom rows). The x axes denote the number of thruster pulses used so far. Each pulse is 1 s long with a 3 s pause following it for a total of 4 s per thruster pulse. The y axes denote the error, 0 meaning a perfect estimate and values above 0 indicating deviation from the ground truth as described in Section 3. The blue solid line shows the error of the estimation of the rotation matrix and the red solid line shows the error of the estimation of the acceleration matrix. All values are averaged from 100 experiment runs. The dashed lines represent error bounds (1 standard deviation).

measured the difference of the learned thruster configuration from the true thruster configuration as the main evaluation metric.

More specifically for each simulation we first chose the thruster configuration at random by sampling the effect of each thruster on the vehicles rotation and acceleration from a uniform distribution. The simulation parameters were selected as follows. We chose the maximum possible rotation speed of the vehicle caused by a single thruster to be 5 rad/s at full thrust. (In comparison we recorded rotation speeds of 2.5 rad/s during rolls effected by a single thruster commanded to operate at half thrust on AMOUR.) We choose the maximum possible acceleration of the vehicle caused by a single thruster to be 0.2 g. (This value was also chosen in accordance with accelerations measured on AMOUR.) For a simulation of a vehicle with N thrusters this resulted in two matrices that described the rotation and acceleration contributions of each thruster:

$$R \in \mathscr{R}^{3\times N} | \text{ s.t. } ||R_{i,1..3}|| \leq 0.05 \text{ and } M_{acc} \in \mathscr{R}^{3\times N} \text{ s.t. } ||R_{i,1..3}|| \leq 0.02$$

During simulation the thrusters were pulsed individually at a manually defined thrust which corresponded to thrust values of $-p$ and $+p$. We performed two sets of simulations, one with $p = 20$ and one with $p = 50$. We pulsed thrusters in order, each first in one direction and then in the other. After all thrusters have been pulsed the simulation continued again with the first thruster and so on. This was done for a total of 50 pulses during each simulation.

For every pulse a thrust vector $T \in \mathscr{R}^N$ with $T_i = \pm p$ where i corresponded to the pulsed thruster and $T_i = 0$ for all other thrusters. We computed the resultant rotation and acceleration by multiplying the rotation matrix R and the acceleration matrix A with the thrust vector T. The computed rotation and acceleration were corrupted by adding a noise value drawn from a normal distribution with mean 0 and standard deviation 0.25 for rotation and standard deviation 0.05 for acceleration. This corresponds to 0.25 rad/s error for rotations and 0.05 g error for acceleration. We determined both values experimentally using AMOUR. The noise represented the average noise over all samples collected during a real world experiment.

For every simulated thruster pulse we store the thrust vector and the rotation and acceleration measurements in the same manner as presented in Algorithm 1. After each pulse we computed a new estimate of the rotation and acceleration matrices R_{est} and A_{est} using the data collected so far. We defined the error as

$$E_R = \frac{||R_{est} - R||}{||R||} \text{ and } E_A = \frac{A_{est} - A}{||A||}.$$

For each thruster number (2,4,6,8) we ran 100 simulations. Figure 2 shows simulation data for the aggregated set of 100 simulations for $k = 2,4,6,8$ thrusters as well as for 4 manually defined configurations. These manual configuration are: (1) 4 thrusters with 2 on each side of the robot, all oriented upwards (2) 5 thrusters in the default configuration as seen in Fig 4(a), (3) 6 thrusters oriented to achieve 6 degrees of freedom (DOF), and (4) 8 thrusters oriented to achieve 6 DOF. The left column

of graphs shows simulation results for $p = 20$ and the right column for $p = 50$. The x axes denote the number of thruster pulses used so far. Since each pulse is 1 s long with a 3 s pause following it this corresponds to 4 s per pulse. The y axes denote the error. The solid lines show the error after a given number of pulses averaged over all 100 simulations. The dashed lines mark the error bounds (1 standard deviation).

The first 4 rows show a gradual decrease in performance. This illustrates that with an increasing number of thrusters the algorithm takes a linearly increasing time to achieve the same estimation accuracy. This is grounded in the fact that the thrusters are pulsed round-robin and so to pulse each thruster a times takes $n \cdot a$ pulses for n thrusters.

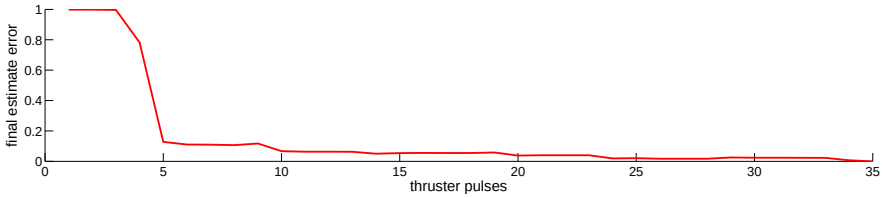
Because the noise is mainly a function of the accelerometer and magnetometer sensor we chose it to be independent of the actual rotational or translational speed of the vehicle. A direct result is that simulations will converge quicker when thrusters are commanded with higher values (faster speed). This behavior is directly visible when comparing the performance of the estimation algorithm in Fig 2(a) ($p = 20$) with the performance in Fig **??** ($p = 50$). All simulations shown assume the same amount of noise, but the estimation accuracy converges much faster to a small value when pulse with a higher thrust are used ($p = 50$). The same behavior was observed also experimentally.

4 Experimental Results

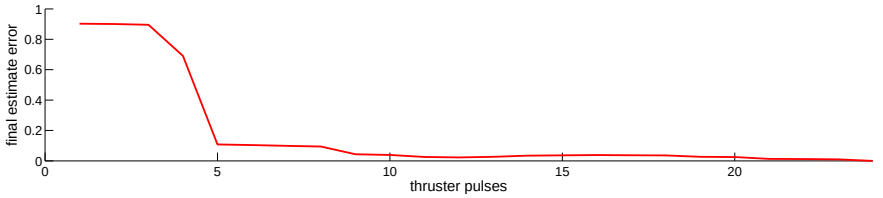
In addition to the simulation studies, we also conducted a set of hardware experiments in the pool using AMOUR [11]. We logged more than 10 hours of experimental data related to estimating thruster configurations. To fit the scope of this abstract we present 3 sets that well reflect the performance of Algorithm 1.

In the first set of experiments AMOUR is configured with the standard five thruster configuration visible in Fig. 4(a). Fig. 6(a) shows that when using the learned parameters the robot can successfully execute and maintain commanded changes in yaw, pitch, and roll. Fig. 6(a) and 6(c) show that the performance obtained by using the learned parameters is similar to that when using manually tuned parameters.

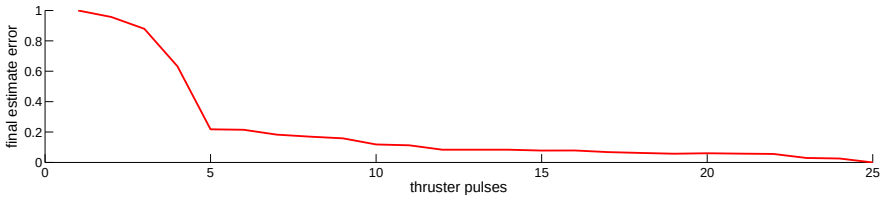
In the second set of experiments we rotate the top yellow thruster by 45 degrees off-axis to simulate a thruster misalignment as seen in Fig. 4(b). We again run the estimation algorithm. Fig. 6(b) shows the performance of the manual parameters for the default thruster configuration. The rotation of the top thruster means that it now has an effect on both pitch and yaw and this is reflected in the yaw error during pitching. Fig. 6(d) shows that the performance of the controller with newly learned parameters. With the learned parameters the robot successfully compensates for the



(a) Estimation convergence for thruster configuration shown in Fig. 4(a).

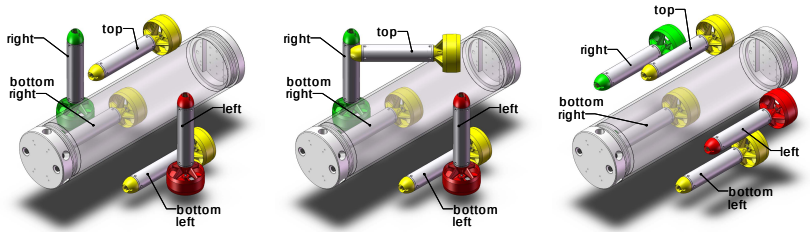


(b) Estimation convergence for thruster configuration shown in Fig. 4(b).



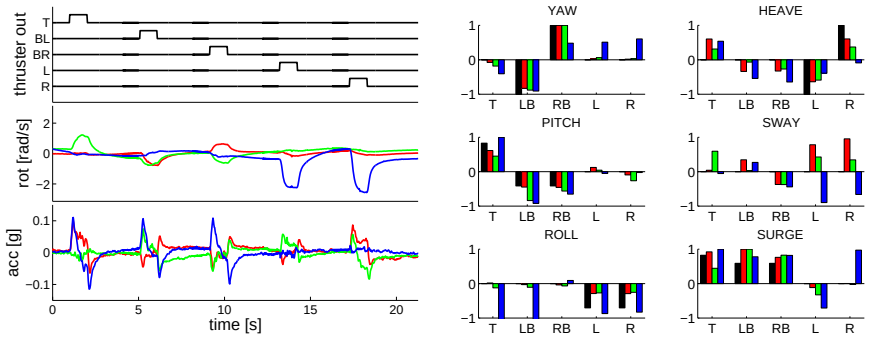
(c) Estimation convergence for thruster configuration shown in Fig. 4(c).

Fig. 3 The convergence of the thruster rotation matrix estimation computed by the estimation algorithm from experimental data. The x axis denotes the number of thruster pulses (for each pulse multiple data points were collected). The y axis denotes the error. Each graph displays the error of estimate E_t when compared to the final estimate E_f for that particular experiment. The error is defined as $error = \frac{||E_t - E_f||}{||E_f||}$. The final estimate was used to compute the error because it was impractical to measure ground truth.



(a) default (b) top rotated 45° (c) all aligned

Fig. 4 The thruster configurations used in the hardware experiments.



(a) Accelerations and rotation changes in response to thruster pulsing.

(b) thruster configs: manual (black), Fig. 4(a) (red), Fig. 4(b) (green), Fig. 4(c) (blue)

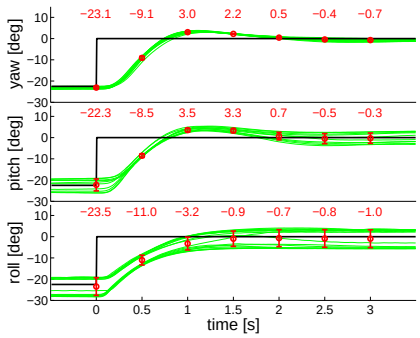
Fig. 5 Experimental data (a) and results of the thruster estimation algorithm (b). In plot (a) the x axis denote time in seconds and the y axis shows: (top) the 5 thruster commands with the thrusters being pulsed successively, (middle) the rotational speed of the robot in response to the thrusters, (bottom) the accelerations as measured on the robot. Red denotes a rotation around or an acceleration along the x-axis of the robots coordinate system. Green represents the robots y-axis and blue the z-axis. In plot (b) the learned rotation matrices and acceleration matrices are shown projected respectively onto the planes of yaw, pitch, roll as well as heave, sway, and surge. The y axis denote the magnitude of a thrusters contribution towards that particular motion. T, LB, RB, L, and R on the x axis denote the thruster as labeled in Fig 4. For each thruster 4 bar graphs are shown: (black) for the manual configuration, (red) for the default robot configuration shown in Fig. 4(a), (green) for the robot configuration shown in Fig. 4(b), and (blue) for the configuration shown in Fig. 4(c).

thruster misalignment and the yaw errors during pitch maneuvers are reduced by more than a factor of 2.

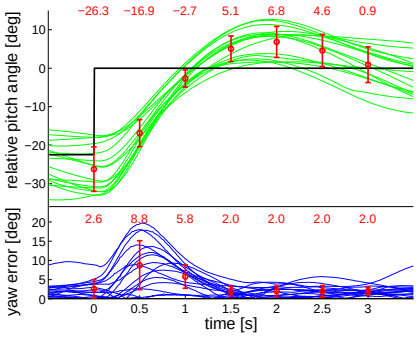
In the third set of experiments all thrusters are aligned with the robot's main body as shown in Fig. 4(c). The blue bars in Fig. 5(b) represent the learned parameters for this configuration. It can be seen that the algorithm correctly recognizes all thrusters contribution to yaw, pitch, and surge maneuvers (the left thrusters screw is left handed, so it's thrust direction is reversed). However it is also visible that the algorithm does not detect missing degrees of freedom and 'learns' to heave, sway, and roll. This sometimes caused oscillations when controlling the robot.

5 Experimental Insights

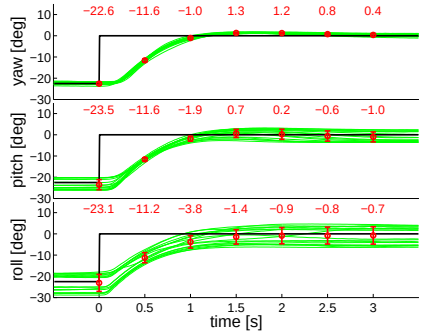
The estimation algorithm eliminates the need for precise manual calibration of thruster positions. It allows quick in-situ estimation of the inverse thruster model, which enables direct control of the robot. In all experiments the algorithm estimates the thruster positions within 40 seconds: only two thrust pulses per thruster, one in each direction, were necessary to learn an inverse model that allowed for control of



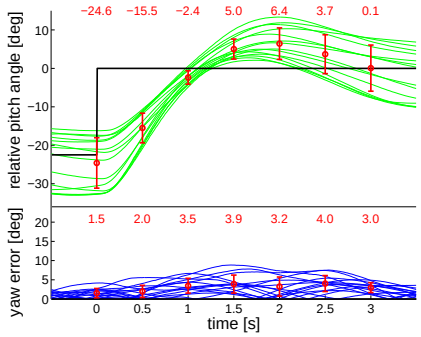
(a) Yaw, pitch, roll performance, standard thruster configuration (Fig. 4(a)), manual parameters



(b) Effect of pitch on yaw, top thruster rotated 45° (Fig. 4(b)), manual parameters



(c) Yaw, pitch, roll performance, standard thruster configuration (Fig. 4(a)), learned parameters



(d) Effect of pitch on yaw, top thruster rotated 45° (Fig. 4(b)), learned parameters

Fig. 6 Experimental results showing performance of robot with thruster positions estimated by the algorithm. In plots (a) and (c) the y axes denote angular attitude of the vehicle projected onto the yaw, pitch, and roll planes. In plots (b) and (d) the y axes denote the pitch angle of the robot (upper part of each graph) and the angular yaw error (lower part of each graph). The black line in all plots represents the desired goal attitude. The green and blue lines represent the robots current attitude and attitude error respectively. The red numbers denote the mean angular error between the robots position and goal.

the robot. The stability of the controller was of equal quality using the estimated parameters and the manual parameters for the default robot configuration (Fig. 4(a)). In the case of a misaligned thruster (Fig. 4(b)) the estimation algorithm correctly identifies the misalignment and the controller outperforms the default manual parameters.

We observed that since the acceleration sensors are noisier than the gyroscopes the translational component of the inverse model was more difficult to learn. Initial

experiments show that learning performance of the translational parameters can be improved by pulsing pairs of thrusters.

We are currently working on methods to determine pulse and pause lengths and when to stop the algorithm automatically. Also, the algorithm does not detect missing degrees of freedom and this can result in erroneous maneuvers when the controller tries to move the vehicle into a direction its thrusters cannot actuate. We are working on an on-line method to detect these failures after estimation and accordingly update the inverse model. Finally, the response time of the controller can be improved by learning a dynamic model of the thrusters that incorporates time-varying behavior.

References

1. Choi, H., Hanai, A., Choi, S., Yuh, J.: Development of an underwater robot, ODIN-III. In: Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, vol. 1, pp. 836–841 vol.1 (2003)
2. Doniec, M., Vasilescu, I., Detweiler, C., Rus, D.: Complete se(3) underwater robot control with arbitrary thruster configurations. In: In Proc. of the International Conference on Robotics and Automation. Anchorage, Alaska (2010)
3. Dunbabin, M., Roberts, J., Usher, K., Winstanley, G., Corke, P.: A hybrid AUV design for shallow water reef navigation. In: Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, pp. 2105–2110 (2005)
4. Farrell, J., Goldenthal, B., Govindarajan, K.: Connectionist learning control systems: submarine depth control. In: Decision and Control, 1990., Proceedings of the 29th IEEE Conference on, pp. 2362 –2367 vol.4 (1990). DOI 10.1109/CDC.1990.204050
5. Gaskett, C., Wettergreen, D., Zelinsky, A.: Reinforcement learning applied to the control of an autonomous underwater vehicle. In: In Proc. of the Australian Conference on Robotics and Automation (AUCRA99, pp. 125–131 (1999)
6. Guo, J., Chiu, F., Wang, C.C.: Adaptive control of an autonomous underwater vehicle testbed using neural networks. In: OCEANS '95. MTS/IEEE. Challenges of Our Changing Global Environment. Conference Proceedings., vol. 2, pp. 1033 –1039 vol.2 (1995). DOI 10.1109/OCEANS.1995.528563
7. Ishii, K., Ura, T., Fujii, T.: A feedforward neural network for identification and adaptive control of autonomous underwater vehicles. In: Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on, vol. 5, pp. 3216 –3221 vol.5 (1994). DOI 10.1109/ICNN.1994.374750
8. Kodogiannis, V.S., Lisboa, P.J.G., Lucas, J.: Neural network modelling and control for underwater vehicles. Artificial Intelligence in Engineering **10**(3), 203 – 212 (1996). DOI DOI: 10.1016/0954-1810(95)00029-1
9. Lorentz, J., Yuh, J.: A survey and experimental study of neural network auv control. In: Autonomous Underwater Vehicle Technology, 1996. AUV '96., Proceedings of the 1996 Symposium on, pp. 109 –116 (1996). DOI 10.1109/AUV.1996.532406
10. Vaganay, J., Elkins, M., Esposito, D., O'Halloran, W., Hover, F., Kokko, M.: Ship hull inspection with the HAUV: US navy and NATO demonstrations results. In: OCEANS 2006, pp. 1–6 (2006)
11. Vasilescu, I., Detweiler, C., Doniec, M., Gurdan, D., Sosnowski, S., Stumpf, J., Rus, D.: Amour v: A hovering energy efficient underwater robot capable of dynamic payloads. International Journal of Robotics Research (IJRR) (2010)

12. van de Ven, P., Flanagan, C., Toal, D.: Identification of underwater vehicle dynamics with neural networks. In: OCEANS '04. MTTS/IEEE TECHNO-OCEAN '04, vol. 3, pp. 1198 –1204 Vol.3 (2004). DOI 10.1109/OCEANS.2004.1405750
13. van de Ven, P.W., Flanagan, C., Toal, D.: Neural network control of underwater vehicles. Engineering Applications of Artificial Intelligence **18**(5), 533 – 547 (2005). DOI DOI: 10.1016/j.engappai.2004.12.004