AMOUR V: A Hovering Energy Efficient Underwater Robot Capable of Dynamic Payloads

Iuliu Vasilescu, Carrick Detweiler, Marek Doniec,
Daniel Gurdan, Stefan Sosnowski, Jan Stumpf and Daniela Rus
Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology

Abstract

This paper describes the design and control algorithms of AMOUR, a low cost autonomous underwater vehicle (AUV) capable of missions of marine survey and monitoring. AMOUR is a highly maneuverable robot capable of hovering and carrying dynamic payloads during a single mission. The robot can carry a variety of payloads. It uses internal buoyancy and balance control mechanisms to achieve power efficient motions regardless of the payload size. AMOUR is designed to operate in synergy with a wireless underwater sensor network (WUSN) of static nodes. The robot's payload was designed in order to deploy, relocate and recover the static sensor nodes. It communicates with the network acoustically for signaling and localization and optically for data muling. We present control algorithms, navigation algorithms, and experimental data from pool and ocean trials with AMOUR that demonstrate its basic navigation capabilities, power efficiency, and ability to carry dynamic payloads.

1 Introduction

Water is essential to many aspects of life. Ocean monitoring and surveillance are very important robot applications. At the moment water monitoring tasks are laborious due to the vast spatial and temporal scale of ocean processes. Even spatial restrictions to observations of micro environments (e.g. the coral reef around an island) are beyond the scope of what biologists can do with manually operated instruments. Over the last two decades, marine sciences have experienced increased use of automated tools for data collection and surveys: sensors with data loggers, remotely operated vehicles (ROV) and autonomous underwater vehicles (AUV). Among these tools, AUVs promise to be the most cost effective and convenient instruments for gathering data. They can perform long, autonomous survey missions using cameras, imaging sonars, and physical and chemical sensors.

This paper presents the design and operation of AMOUR V (Autonomous Modular Optical Underwater Robot version 5), an AUV tailored for operations in coastal waters and reef environments. Its applications include marine biology studies, environmental, pollution and port security monitoring. The main design requirements for our robot have been: (1) a small, light-weight package that can be carried by a single person and fit in a suitcase; (2) low-cost modular configuration; (3) ability to hover; (4) ability to pick-up, travel with and drop off payloads; (5) ability to interact with an underwater sensor network; and (6) ability to carry standard marine instruments over medium range missions (10km). We considered the option of using and adapting an existing underwater robot design, but decided that this path would not easily satisfy our design requirements. We opted to design a new robot system from scratch in order to address the requirements in an organic integrated fashion, as opposed to an add-on approach.

AMOUR V is a cylindrically shaped, 25kg, closed haul, hovering robot, capable of maintaining stable

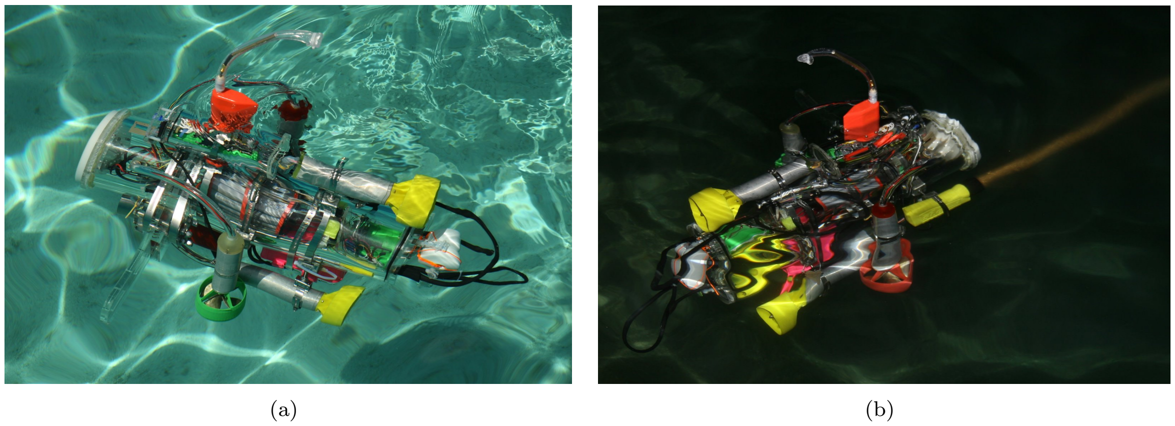


Figure 1: Amour V deployed (a) for a daylight mission and (b) for a night mission

position and rotation[1]. It is statically and dynamically stabilized in the pitch and roll axes. Its actuation consists of 5 thrusters, a balance control mechanism, and a buoyancy control mechanism. During its mission the robot can easily switch between vertical and horizontal orientation depending on the task. A key capability of Amour V is the ability to operate in symbiosis with a wireless underwater sensor network. Amour V is fitted with a docking mechanism which enables the robot to autonomously pick up sensor nodes for deployment, relocation and recovery. In order to transport efficiently the dynamic payload of the sensor nodes, the robot was fitted with internal buoyancy and balance control. Amour V relies on a water piston mechanism to compensate for changes in buoyancy of up to 1kg due of payload. In addition an internal battery moving mechanism enables the robot to change its center of mass position and accommodate the same 1kg payload at its tail end without help from the thrusters for balance. The buoyancy and balance control systems provide an efficient way for controlling the robot's buoyancy, center of mass and overall orientation. Both of these systems save significant energy during the robot's mission as they require energy during adjustment only. In the absence of buoyancy and balance control mechanisms the robot would have to use the main thrusters (and hence a large amount of additional energy) to compensate for the payload. Our long term goal for this type of underwater robot is complex autonomous and collaborative underwater pick and place operations–for example assembling underwater structures, collecting samples or using tools for repairs.

Amour V operates in connection to a sensor network. The robot and the sensor network nodes are fitted with custom developed acoustic and optical modems for low data rate broadcast and high data rate point-to-point communication respectively. The sensor network also acts as a GPS-like constellation of satellites, providing the robot with precise position information. This eliminates the need for the robot to include large expensive underwater position information sensors for navigation, such as Doppler velocity loggers or laser ring gyros. The synergy between the network and the robot goes even further. Amour V can visit the sensor nodes and download their data using its high speed, low power optical modem, enabling real time, in-situ data muling, which is not practical with acoustic telemetry and without using mobility.

The capabilities of networked underwater sensors and robot systems open new doors for marine biology and environmental studies. Consider a study on the effect of phosphates and nitrates on coastal reef health. The sensor nodes (fitted with the relevant chemical sensors) will be dropped the area of interest. The nodes will establish an ad-hoc network, self-localize and start collecting the data. When significant chemical activity is detected, the network will signal the base through acoustic multi-hop telemetry. The robot will be guided by the network to the event location for a complete chemical survey and imaging of the coral heads

[1] Amour V plus a network of 10 sensor nodes can be easily fitted in two hard cases within the airlines' size and weight limit.

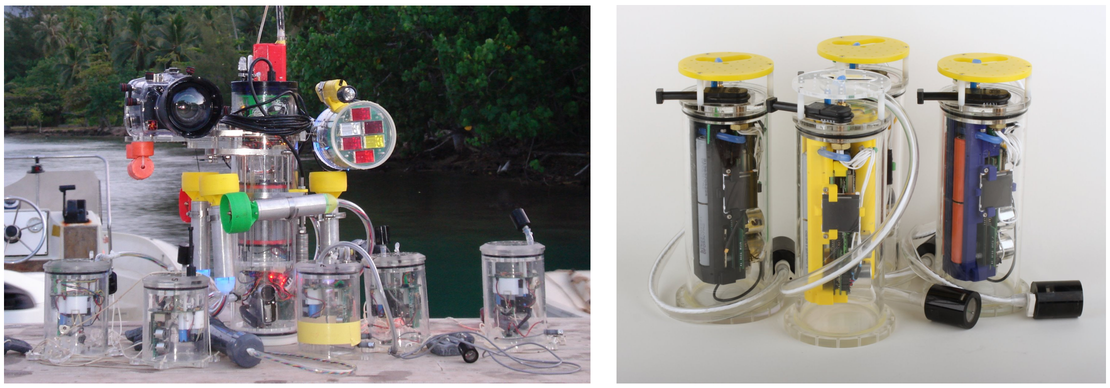

(a) (b)

Figure 2: (a) Amour V with an imager payload and the sensor network nodes. (b) the latest generation of sensor nodes, smaller and lighter.

(for health assessment). Using acoustics, the robot will use the network nodes as beacons for navigation and localization. Additionally the robot can visit each node individually and download the collected data through the short range high speed optical link. The essential robot capabilities for this task are hovering and maneuverability. Following the analysis of the muled data, the robot will be able to reposition the sensor nodes for enhanced sensing. In a second mission it will dock, pick up and relocate the sensor nodes from the chemically inactive areas to the more dynamic areas, increasing the sensor density. These activities can be repeated over many months with very little human interaction. For the duration of the study, the scientists will be provided with a quasi real time stream of data. This will enable quick detection of sensor failure, procedural errors and adaptive data collection. At the end of the study the robots will autonomously retrieve the sensor nodes and return the sensor nodes to dock.

1.1 Related Work

Our work builds on a large body of previous research in underwater robotics. The field of underwater autonomous robots started in early 1990s with robots such as Woods Hole's ABE and MIT's Odyssey. Since then the field has undergone continuous technological improvement. We briefly survey the related work in underwater robot navigation, buoyancy control, balance control, and localization. For a survey of AUVs and ROVs see Yuh and West [37] and Antonelli *et al.* [2].

Today's AUV landscape includes several companies such as Hydroid and Bluefin Robotics that produce AUVs as their main product. These products are torpedo shaped robots optimized for long surveys with sensors. The robots are unable to hover (as they need to be in constant motion). These AUVs include the Remus series from Hydroid [29] and the Bluefin series from Bluefin [1]. These robots were developed primarily for mine detection and mapping, but have also found wide use in the research communities.

The research community has also developed a number of non-hovering AUVs. SAUV [5] has solar cells which allow it to recharge at the surface for long endurance missions. The STARFISH AUV [25] was designed to be modular and easily upgradeable.

The ability of an underwater vehicle to hover is commonly found in remotely operated vehicles (ROVs). Autonomous vehicles with hovering abilities include AQUA [9] which is an amphibious robot with six flippers and Finnegan the RoboTurtle [18] which is a biologically inspired four flipper robot. The ODIN-III [3] has eight thrusters and a one degree of freedom manipulator. The Seabed [27] AUV is designed for high resolution underwater imaging. It has a dual torpedo shape with enough degrees of freedom to support hovering. The WHOI Sentry AUV [19,36] (the successor to ABE) is designed for efficient cruising but is also able to rotate its thrusters to hover for near-bottom operations. The WHOI Jaguar and Puma AUVs [16] were developed to study the ocean under the Arctic ice cover. CSIRO's Starbug [10] can hover, however it is most efficient while moving forward. The HAUV [32] is an ROV-like vehicle with the ability to perform autonomous operations such as ship hull inspection. Amour introduces a modular composable thruster design that enables navigation and hovering in both vertical configuration (for high maneuverability) as well as in a more streamlined horizontal configuration (for long duration missions.)

Most underwater robot systems are neutrally buoyant and do not execute tasks that require control of their buoyancy. Underwater robots that can change their buoyancy include Divebot [20] which changes its density by heating oil, SubjuGator [17, 23], which uses two solenoids which regulate the amount of ballast in a buoyancy compensatory, and gliders such as the Spray gliders by Bluefin Robotics [26] or University of Washington's Sea Gliders [13] which change their buoyancy by battery-powered hydraulic pumps in order to glide forward. Our work provides simultaneous control of balance and buoyancy to enable the pickup of a payload. Gliders are the only class of robots that also control both balance and buoyancy. However, gliders do so with different mechanisms to power their movement and are not able to pick up objects.

Navigation and position estimation for AUVs is typically achieved by using a combination of a Doppler velocity logger (for dead reckoning) and laser ring gyros (for precise orientation). These sensors are far too large and expensive for the scale of Amour. Our robot instead relies on an external localization system provided by a statically deployed sensor network. The network self localizes and provides localization information to our robot [6, 8, 35]. This differs from the traditional long base line (LBL) systems which is how many AUVs localize [22, 24, 31]. Our system does not require a priori localization of the sensor nodes. Additionally, our system receives non-simultaneous ranges (since the robot and sensors also transmit data) requiring a different algorithm than traditional LBL networks. See [6] for more detail.

This paper presents the first comprehensive description of our underwater robot system. Components of this system have been described in greater detail in M.S. theses and conference proceedings. The first design iteration of this robot was reported in [34]. The robot's low level controllers, IMU, and balance and buoyancy mechanisms have been the subject of several M.S. theses [15, 28, 30]. Initial experiments with balance and buoyancy control have been reported in [7]. Details on docking and data muling can be found in [12, 33] and the algorithms for docking with another robot and performing cooperative navigation are described in [11]. The external localization algorithm and experiments are presented in [6, 8, 35].

1.2 Outline

The remaining of this paper is organized as following. Section 2 presents the hardware description of the Amour V. Section 3 presents the control and sensor data processing algorithms, including balance and buoyancy compensation. Section 4 presents the data from our pool and field experiments with Amour V. We present our conclusions from building and operating the robot in Section 5.

2 Hardware Description

Our underwater system has two components: the robot and the underwater sensor network system. The robot incorporates a sensor network node in its configuration which allows us to treat it as a mobile node in the sensor network system. The system is modular and the various components of the robot and the sensor network nodes can interact in different configurations.

2.1 Amour V Architecture

Amour V's body is built from a 63cm long, 16.5cm diameter acrylic tube with a wall thickness of 0.6cm. The cylindrical body shape was chosen for hydrodynamic reasons. The body is a low cost, efficient pressure vessel. In order to dock and carry a sensor node efficiently, the robot was designed to be operational with the body both in horizontal and vertical configurations. The horizontal configuration is used for long distance, streamlined travel. The vertical configuration is used for docking and data muling. Vertical docking allows a sensor node to be attached at the end of the robot body, thus maintaining the streamlined shape during travel. The robot is modular. It is easy to add or remove modules such as sensor network nodes, sensing modules, as well and the buoyancy and balance control modules.

Attached to the top cap of the tube is the *head* of the robot. It contains the electronics for planning and control (see Section 2.3) and the Inertial Measurement Unit IMU (Section 2.3). The head is contained in a splash proof container for easy removal in-situ or immediately after the mission. The mid section of the body contains the battery module supported by 4 stainless steel rails. The balance control mechanism sets the battery module's position along the length of the robot (Section 2.5). The battery module is composed of 72 rechargeable Lithium Ion cells with a nominal voltage of 3.7V and a charge of 2Ah each. We used readily available 18650 laptop cells. The battery is organized as a 4-series cell with a nominal voltage of 14.8V and a charge of 36Ah, thus a total energy capacity of over 500Wh.

Below the battery module there is the power PCB, which contains the power distribution and voltage conversion circuits. The space between the power PCB and the lower cap can be used for additional payloads (e.g. a video camera). The lower cap of the robot is replaceable with the buoyancy and docking module (see Section 2.5) or other modules. The two end caps of the main tube are held in place by lowering the inside pressure of AMOUR V's body below atmospheric pressure. We operate the robot with inside absolute pressure of 0.9Bar, which creates a holding force on the end caps of about 180N at sea level, and increasingly higher as the robot dives. The end caps are sealed using standard o-rings. An internal pressure sensor can easily identify potential leak, before deployment of the robot in water.

AMOUR V has five thrusters (Section 2.2) attached to the main body using easily removable stainless steel clamps. The thrusters are electrically connected to the main body through flexible PVC tubes, which provides an effective and significantly less expensive solution than underwater connectors. Three thrusters are collinear with and evenly distributed around the main body tube. The remaining two thrusters are orthogonal to the main body. This thruster configuration allows the robot to be oriented and controlled in both horizontal and vertical configurations. Additional thrusters can be added to the robot. Previous revisions of the robot included only four thrusters.

In the horizontal configuration the robot is streamlined for energy efficient motion. In the vertical configuration the robot is more maneuverable. This configuration is used when docking with sensor nodes or downloading data via the optical modem. In both configurations the robot has independent control over its pitch, roll and yaw angles, as well as its depth and forward/reverse motion (details in Section 3.2). Side motions are not possible in either configuration. In the vertical configuration the robot is able to spin along the yaw axis very rapidly mitigating the lack of direct lateral control. This allows the robot to respond to currents and unexpected water motions. In the horizontal configuration rotation is slower. This configuration is used for long distance travel when maneuverability is less important.

In addition to the thrusters, the robot can carry additional sensors and devices attached to its body: video camera, still camera, scanning sonar, range finder, and/or one of our sensor nodes. We run the robot with one of our network sensor nodes attached to its body. The node's CPU runs the high level mission control. In addition the sensor node contains a radio modem and GPS (for surface link with the robot), acoustic and optical modem for underwater comms, an SD card for data logging, temperature sensor and 24bit ADCs for additional survey sensors (e.g. salinity). When the robot travels in horizontal configuration close to the water surface, the GPS and radio antenna can stay above water for localization and mission control.

2.2 Thrusters

We opted to design and fabricate AMOUR V's thrusters as the commercially available units were too big, did not have the required thrust, and were generally very expensive. The thrusters are built around Himax HB3630-1000 sensorless brushless DC motors fitted with a 4.3:1 planetary gear box. The motors are rated for 600W and are intended for the RC market. They have a very low winding resistance and thus are able to generate high power efficiently in a small package (with proper cooling). Brushless motors were chosen for their higher power and efficiency compared to the mechanically commutated DC motors. The placement of the windings on the stator (the body) is also advantageous, as efficient cooling can be easily insured by thermally connecting the motor's body to the thruster case.

Each thruster is driven by an in-house developed brushless motor controller built around an LPC2148 processor and custom power electronics. The controller is located inside the thruster and is attached to the back of the motor. The controllers receive commands from the main CPU over a RS485 bus and regulate the motor speed accordingly.

The motors, gearbox and electronics are housed in a watertight aluminum case. The propeller's 6mm stainless steel shaft penetrates the top cap and is sealed by a double o-ring. Between the two o-rings there is an oil cavity that assures lubrication and prevents overheating. The propeller is a Prop Shop K3435/4 with a 90mm diameter and 100mm pitch. The propeller size was determined experimentally to maximize the static thrust and efficiency. The propellers were statically balanced in house (by removing material from their blades) to reduce vibrations and improve efficiency.

2.3 Central Controller Board

Amour V's motions are coordinated by the Central Controller Board (CCB). The CCB receives commands from the sensor node high level mission control and sensory information from the IMU. The CCB sends commands to the 5 thrusters, buoyancy module and balance module. The CCB is responsible for running the low level control loops, maintaining the commanded configuration, attitude, depth and speed.

The CCB is built around an 32bit LPC2148 ARM7 processor running at 60Mhz for computation. A Cyclone II FPGA is used as communications co-processor. All serial links between the LPC and the rest of robot's subsystems (thrusters, buoyancy/balance, IMU, sensor node) are implemented in FPGA. All the serial links are buffered inside the FPGA, This greatly reduces the CPU time necessary for communications, and frees cycles for control loop calculations.

2.4 Inertial Measurement Unit

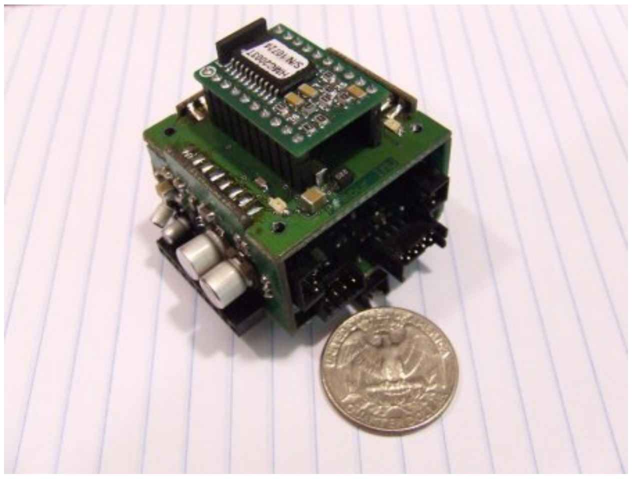


Figure 3: Inertial Measurement Unit of Amour V

The IMU (Figure 3) is used to determine the absolute orientation in pitch, roll and yaw of Amour V. The robot is neutrally balanced and consequently can float in a random orientation. The IMU uses 10 sensors to measure Amour V's attitude and depth: three orthogonally mounted acceleration sensors, three solid state gyroscopes, three magnetometers and one pressure sensor. The accelerometers (ADI ADXL103/ADXL203) are used to determine the direction of the gravitation vector, and thus the absolute orientation of the robot to vertical direction. The accelerometers provide an absolute measurement but they are noisy and prone to biased values due to the robot's acceleration during motion.

The gyros (ADI ADXRS300) are used to measure the rate of turn on all axes. Their readings are integrated to provide orientation and are thus prone to drift over time. However their reading are robust relative to accelerations and have very good signal to noise ratio. The magnetic sensors (Honeywell HMC2003) provide the direction of the Earth magnetic field. The IMU uses the magnetic vector and the orientation of the robot to compute the robot's tilt compensated heading. The pressure sensor (Freescale MPXA4250A) measures the depth, ascent and descent rates. All the sensor measurements are low-pass filtered (MAX7401) converted to digital values using a 16-bit A/D (ADS8344). A CPU (LPC2148) fuses the raw sensory data and computes accurate orientation angles and depth of the robot (Section 3.1). Figure 4 shows a block diagram of the connections in the IMU. All the IMU data is sent to the Central Controller Board (CCB) via a RS485 serial data link with an update rate of 100Hz.

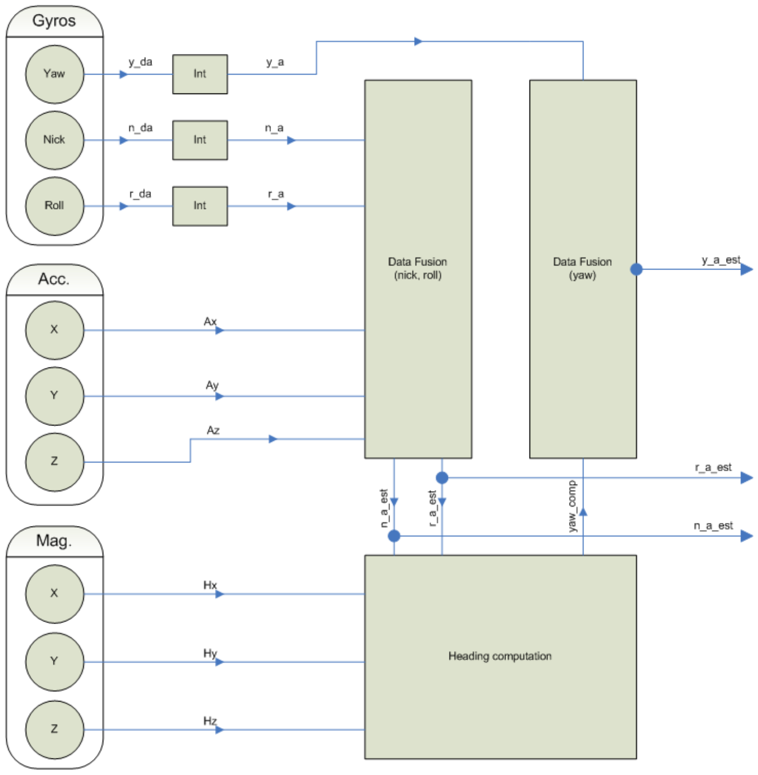


Figure 4: Inertial Measurement Unit Overview

2.5 Buoyancy and Balance Mechanisms

We extended the robot with buoyancy and balance control mechanisms, packaged as auxiliary modules (see Figure 6). The design requirement for the mechanisms has been to achieve adaptation to an additional payload of up to 1 kg within 30 seconds. The robot uses its thrusters to achieve a desired depth and orientation. Adding extra weight to the robot causes the thrusters to expend more energy. The role of the buoyancy control module is to bring the robot back to a neutrally buoyant state (and thus save energy). The role of the balance control module is to change the center of mass of the robot when additional weight is added or in response to the change in the buoyancy control module.

The buoyancy mechanism (see Figure 5) controls the buoyancy of the robot by moving a piston inside a

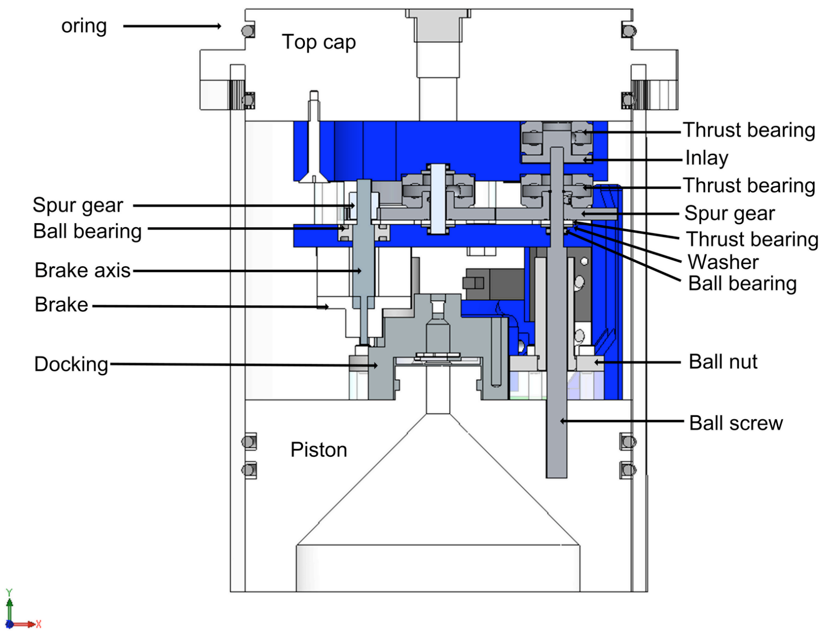


Figure 5: Diagram of the buoyancy control mechanism.

cylinder. The cylinder has a 16.5 cm diameter. The piston can travel 5.6 cm over 24 sec and has 1.2 kg lift capacity. The effect on buoyancy is $\Delta m = A\rho\Delta h$. The buoyancy system coordinates with the balance control mechanism (see Figure 9), which alters the center of mass of the robot by moving the battery pack up and down in the robot. The buoyancy system has an integrated docking mechanism (see Figure 5 and [11]) which enables the automatic pickup of payloads that are compatible with the docking mechanism (for example, the underwater sensor network nodes can be picked up by this robot). The buoyancy control module is contained in a water-tight cylindrical tube that can be attached to the robot with an underwater connector. It includes a piston moved by a set of three ball screws with ball nuts. A custom-designed gear box ensures that the ball screws are turned simultaneously and provides a gear ratio that can compensate for forces arising at up to 40m depth[2]. The output power P_{out} of the motor is $P_{out} = \frac{(V - k_r \frac{T_m}{k_m}) k_s T_m 2\pi}{60}$, where V is voltage, T_m is the motor torque, k_r is the terminal resistance in Ω, k_m the torque constant in $\frac{mNm}{A}$ and k_s as the speed constant in $\frac{RPM}{V}$. We found the best performance for gears of 64 teeth for the ball screws and 32 for the motor gear.

The balance control module is implemented as a moving battery module inside the robot. This module acts as a movable weight of 5.7 kg. Shifting this weight inside the body of the robot (full time takes 30 sec) changes the center of mass. The battery system moves by sliding on four rails. Its position is controlled with a 20 turns per inch lead screw connected to a motor over a worm gear with a total 11:1 reduction. Figure 9 shows the different states of the robot with and without attached weight. The balance control system is not back-drivable and can compensate for up a weight of 1 kg attached to the tail of the robot. The balance system is especially important for maintaining the robot's angle when traveling in horizontal configuration.

[2]The force on the piston is about 10500N at 40m depth.

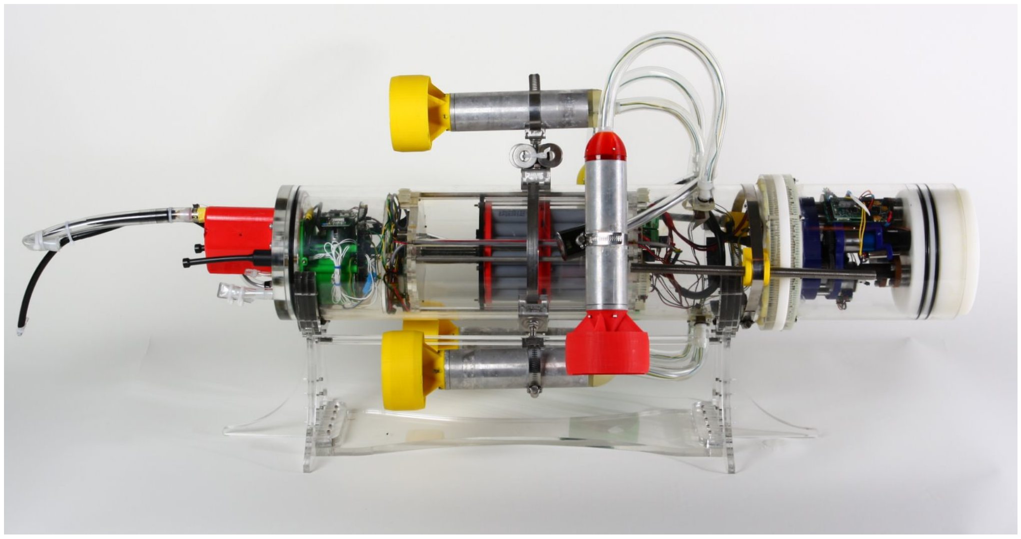


Figure 6: AMOUR V with the buoyancy and docking module attached (right side). The balance mechanism is visible in the center of the body. The battery pack can be moved left or right using the center lead screw.

2.6 Docking Mechanism

The underside of AMOUR V's buoyancy module is a conical shaped cavity that allows docking with any object that holds a docking element — a 15cm long rod of 1cm diameter with a modulated light source at its base as beacon. AMOUR V uses 4 optical sensors distributed around the cavity circumference for optical alignment during docking procedure. The conical shape of the docking cavity provides the mechanical compliance during the final stage of docking. This alignment mechanism is similar to the probe and drogue mechanism used by NASA's Apollo program. A latch plate with a variable width hole located at the apex of the cone is actuated by a servo-motor in the horizontal plane in a tub-like compartment. To dock, the plate is actuated such that the smaller side of its variable width hole is latched around a corresponding groove on the docking rod. This creates a strong link between the robot and the docked object (e.g sensor node) capable of withstanding forces of up to 200N. For additional details on the docking mechanism see [34].

2.7 Sensor Node

The robot's high-level mission control, communications and sensing are performed by a separate computational unit. This component is a unit of the underwater sensor network system. Thus, the robot can be viewed as a mobile node in the underwater sensor network.

The sensor node is built around a 32bit ARM7 LPC2148 CPU running at 60Mhz. The node is capable of multi-modal communications and has several standard sensors.

For communication the node contains a radio modem, Bluetooth, acoustic modem and optical modem. In software, physical communication layers are abstracted, and therefore all types of communications are possible on each of the physical layers. In practice however, due to the vastly different performance and required conditions, the 4 modes of communication are used for specialized purposes.

The radio modem (900Mhz 1Watt Aerocomm AC4790-1000) is used when the robot is on the water surface for uploading the mission, setting parameters or transmitting commands (e.g start mission). The main advantage is the long range (aprox. 200 meters max observed range) and good data rate (76kbs), which gives reliable communication while the robot is operating near the boat. The Bluetooth module is used for uploading new software or software updates in the sensor node and downloading data logs. It is convenient as it requires no wire, no underwater connectors and no opening of the node. Bluetooth supports a high data rate (1Mbps). Its disadvantage is the limited range (10 meters).

The acoustic modem is used for underwater communication (primarily in broadcast mode), signaling, ranging and localization. It is an FSK modem, operating at 30Khz with a data rate of 330bits/sec over 400 meters of range. The modem was developed in-house. It provides a platform for experimenting with low level modulation schemes, MAC protocols, and ranging algorithms. The acoustic modem supports the measurement of pair-wise ranges (using time of flight), which, are used to establish a system of coordinates and to locate and track objects in this system of coordinates. One of the goals for the acoustic modem has been to avoid the high cost and closed source nature of commercially available modems. The modem is built around a BF533 Blackfin DSP and contains all the electronics needed to drive and receive signals from a piezoceramic transducer. The modem implements a TDMA system for medium access. The network nodes communicate during their designated time slots which are determined adaptively, according to the number of nodes in the network. The modem includes a thermo-compensated clock, which enables time-of-flight one-way ranging. In turn, ranging enables robot localization without additional overhead on the network bandwidth. For more details see [4, 8, 35].

The optical modem provides high-speed short-range point-to-point communications underwater and on the water surface. It operates on 530nm (green light) and has a max speed of 1Mbps over a range of 3 meters. Underwater it is used for close range communication between the robot and the static sensor nodes (data muling). On surface can be used for reprogramming or data downloading. For more details see [12].

In addition to communication, the sensor node contains a GPS module (usable while the robot is close to the water surface), a pressure sensor, and a temperature sensor (with millidegree resolution). Additional sensors can be easily connected to the internal 24bit A/D. A mini-SD card is used for data logging.

3 Control

In this section we describe the algorithms used by the robot to navigate and to carry payloads. The robot uses its actuators to actively maintain the desired pose based on the sensory input from the IMU, pressure sensor, localization network and GPS (when on surface).

3.1 Pose estimation

The robot uses the IMU to measure the body's absolute orientation in Roll, Pitch and Yaw relative to the Earth's coordinate system (defined by the gravitation and magnetic field). The three types of sensors used are: acceleration sensors to measure static as well as dynamic accelerations, gyroscopes to measure rotations around the IMU's internal X, Y and Z-axis, and magnetic field sensors to measure the earth's magnetic field.

The acceleration and gyroscope measurements are fused in order to get accurate measurements of the roll and pitch angles. The Kalman Filter is the optimal linear estimator for this type of problem [14], however, the matrix-operations required for a Kalman-Filter are computationally complex requiring significant CPU time. To achieve higher update rates for reduced latency, the IMU uses a simple and efficient algorithm presented in Algorithm 1.

Algorithm 1 Pose Estimation

```
1: read the accelerometers on all three axes X, Y, Z
2: read the gyroscopes on all three axes X, Y, Z
3: compute $(Roll_t^{(g)}, Pitch_t^{(g)}) = (Roll_{t-1}, Pitch_{t-1}) + \int gyroscopes$ the new gyro pose estimation
4: compute $(Roll_t^{(a)}, Pitch_t^{(a)})$ based on accelerometer readings
5: compute $(Roll_t, Pitch_t) = \alpha(Roll_t^{(g)}, Pitch_t^{(g)}) + (1-\alpha)(Roll_t^{(a)}, Pitch_t^{(a)})$, the new pose estimation by
   reconciling the gyro and accelerometer readings
```

The algorithm computes the pose estimation as a weighted average between the gyroscope's estimation (integration of the rate of turn) and accelerometer's estimation (assumes the acceleration is vertical, caused just by gravitation). The weighting factor α was chosen close to one (i.e. heavy weight on gyro estimation) due to the immunity of gyro readings the to robot's accelerations. The accelerometers estimation is used only to prevent the estimation drift due the gyro bias and integration.

Using the reliable estimate of the current pitch and roll angles stored in *angle_pitch* and *angle_roll* the heading is computed using the magnetic vector projection described above.

Simple functions using only fixed-point-numbers, lookup-tables and linear interpolation were programmed to do the computation of all trigonometric functions in a reasonable amount of time. These functions can be executed in about 600 cycles per second in the IMU's microcontroller.

3.2 Hovering and Motion

AMOUR V can hover and navigate in two orientations — vertical and horizontal — denoted by the robot's body orientation. In both orientations the pose is actively maintained using the thrusters. In the horizontal configuration, the two thrusters perpendicular to the body are used for controlling depth and roll, while the three thrusters parallel to the body are used to control pitch, yaw and forward/backward motion. In the vertical configuration, the two thrusters perpendicular to the robot's body are used for controlling yaw and forward/backward motion, while the three thrusters parallel to body are used for controlling pitch, roll and depth. In both configurations the robot is fully actuated, therefore pitch, roll, yaw, depth and forward/backward motion can be controlled independently.

Control Output	T0	T1	T2	T3	T4
Pitch	0	0	-1	$+\frac{3}{4}$	$+\frac{3}{4}$
Roll	0	0	0	+1	-1
Yaw	+1	-1	0	0	0
Depth	0	0	-1	-1	-1
Forward Speed	+1	+1	0	0	0

Figure 7: Control output mapping for vertical orientation of the robot

Control Output	T0	T1	T2	T3	T4
Pitch	0	0	-1	$+\frac{3}{4}$	$+\frac{3}{4}$
Roll	-1	+1	0	0	0
Yaw	0	0	0	+1	-1
Depth	+1	+1	0	0	0
Forward Speed	0	0	+1	$+\frac{3}{4}$	$+\frac{3}{4}$

Figure 8: Control output mapping for horizontal orientation of the robot

The CCB uses the robot pose estimates computed by the IMU to control the pose. The pitch, roll, yaw and depth controllers are very similar. They are PD loops operating on the difference between the desired pose and the current pose estimated by the IMU. Figure 11 presents the depth and pitch controllers. The outputs of the PD controllers are used for thrusters commands and as an input for the buoyancy and balance mechanisms (Section 3.3). The roll and yaw controllers are similar, but their output is used only for commanding the thrusters. The four controllers are run independently, and their output is mixed for the correct thrust depending on the robot's orientation. Table 7 and Table 8 present the complete controllers' output mapping to the robot thrusters.

The values for the PD loop gains were determined experimentally. Compensations were added for the asymmetric thrust generated by the propellers (i.e. for the same rotation speed the propellers generate twice the forward thrust compared to back thrust).

Switching between the horizontal and vertical modes of operation is described by the Algorithm 2. The robot starts with a request for a change in pitch which is followed by the thruster mapping that corresponds to vertical or horizontal configuration as appropriate.

Algorithm 2 Switch between horizontal and vertical orientation

```
1: request Pitch angle of $45°$
2: wait till the AMOUR V's Pitch angle is $45°$
3: switch to the mapping table corresponding to the desired orientation (Table 7 or Table 8)
```

For navigation between waypoints, the robot maintains roll and pitch angles of zero degrees and the specified depth profile. The heading is computed as a direct vector between the current position estimate and the desired waypoint. The speed is set as specified by the mission and mapped into thruster output based on the current orientation (Table 7 or Table 8).

3.3 Buoyancy and Balance

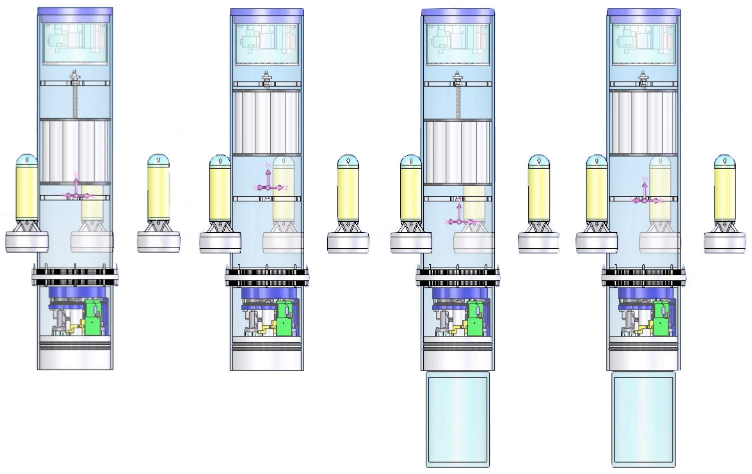


Figure 9: Simulation of the center of mass location. Robot without weight, battery down. Robot without node, battery up. Robot with weight, battery down. Robot with weight, battery up.

Suppose a payload of mass m (in water) is attached to the robot at depth d and carried for time t. If the robot uses thrusters only, the energy consumption is: $E = K_p \times m^2 \times t$, where K_p is the a constant related to the thrusters efficiency (which is about $20W/kg^2$ in our system). The longer the payload is carried the more energy it uses. If instead the buoyancy engine is used the energy consumption is $E = K_b \times m \times d$, where K_b is a constant related to our buoyancy engine efficiency (which is about $35J/(kg*m)$ in out system). Figure 10 shows the trade-offs between using thrusters only and using thrusters and a buoyancy engine. The energy does not depend on the time the payload will be carried, but does depend on the depth at which is picked up. For example, for a 1kg payload collected at $10m$ depth the thrusters will use $20J/sec$ while the buoyancy engine will use $350J$ so the break even point is 17.5 secs. For the first 17.5 seconds the thrusters are more efficient. If, however, the object is carried for 5min, the thrusters will use 20 times more energy than the buoyancy engine.

Figure 11(a) shows the control loop for the depth and buoyancy engine. The thrusters are controlled by a PD feedback loop that corrects for the desired depth. The buoyancy engine receives a low pass filtered version of the thruster command as input for a PID feedback loop that controls the piston's position. The buoyancy engine moves the piston in the direction that brings the robot to neutral buoyancy, close to 0 percent thruster output.

Figure 11(b) shows the control loop for the battery engine. The thrusters are controlled by a PD feedback loop that corrects for the desired pitch. The battery motor receives a low pass filtered version of the thruster command. The battery is moved in the direction that brings the robot to neutral balance by a PID feedback loop that controls position.

- Thrusters

$$E = K_p \times m^2 \times t$$

- Buoyancy

$$E = K_b \times m \times d$$

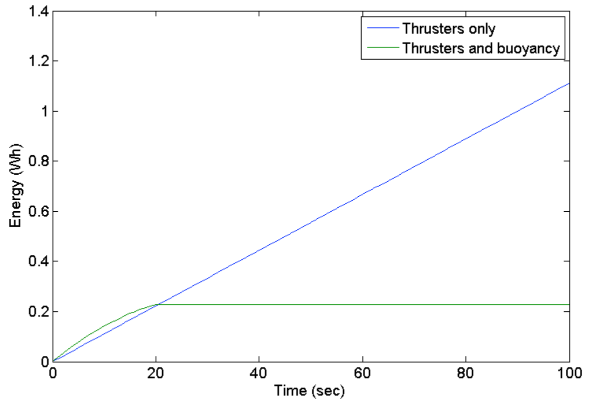


Figure 10: Energy Analysis for a system that uses (a) thrusters (top curve) only and (b) thrusters and buoyancy engine (bottom curve). The x-axis shows the system's energy consumption in Wh. The y-axis shows time in sec. The simulations were done for a payload of 1kg at 5m depth. The constants used were $Kp = 40W/(kg^2)$ and $Kb = 160J/(Kg * m)$.

3.4 Localization and Tracking

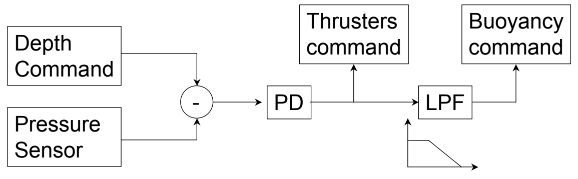
While at the surface the robot can use GPS, however, once it is under the water, GPS and other radio devices are not available. We use the acoustic modems to communicate and obtain ranges from the robot to the static nodes underwater. The main challenge in localizing the robot using ranges to static nodes is that the ranges measurements from different nodes are not received simultaneously. The robot can move significantly between range measurements (order of meters in our system).

3.4.1 Communication

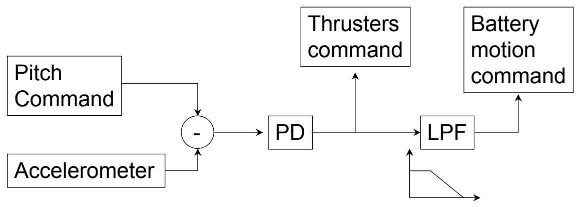
Before the nodes can localize themselves they must decide on a communication schedule. In our system this is done using a self-synchronizing time division multiple access (TDMA) scheme to schedule messages.

Each network deployment starts with initializing the number of slots at deployment type. The system starts in a configuration where the number of slots equals the number of sensor nodes. We can add slots on the fly at any time. The additional slots are allocated to support more frequent communication for the moving nodes, or for base stations. Nodes are allocated a slot number at the beginning of the deployment. However, any node can send a command to another node to release its time slot. Our TDMA protocol uses 4s time slots. Each time slot is divided into a 2s master packet for the slot owner, and 2s response time. The response time may be one of two categories, based on the owner's request. The owner may request communication to a single node, or communication to multiple nodes. In the case of communication to a single specific node, the response includes data. However, the response packet is also used to compute the range between the two nodes (e.g., the roundtrip time offset by 2s.) In the case of multiple destinations, the responses are spaced at 200 ms intervals and are used primarily to compute multiple ranges within one communication slot. This communication modality does not support much data transfer, but it enables a very efficient way of estimating multiple ranges. This feature is important for the tracking system. It allows the robot to obtain multiple ranges to the static sensor nodes in quick succession.

In our TDMA implementation, a node can own anywhere between 0 and N time slots. The owners of the slots can be changed dynamically in real time. A key feature of our sensor network system is the ability of the system to self-synchronize without access to an external clock source such as a GPS or to very high-precision clocks. The self-synchronization algorithm works as follows. The nodes are initialized with the total number



(a)



(b)

Figure 11: The control loops for the buoyancy and balance systems.

of slots. Each node knows its slot number. When node `n` with allocated slot `N` is deployed in water, it waits to hear a correct master packet for `4s`. If the node hears a message in this interval, it decodes the slot number from the message and computes its own time to talk based on this slot number. So, for example if `N = 4` and `n` hears a message from the node whose allocated slot number is 3, node n knows that its time to communicate is 4 seconds later. This method allows nodes to synchronize their internal clocks. If node `n`, allocated slot `N`, does not hear anybody during its first `4s`, `n` starts transmitting messages. Eventually, one of the nodes will start communicating first and all the other nodes will synchronize to it. A possible deadlock happens if all the nodes start talking at once. This is a low-probability event. During over 100 trials, we have never encountered this situation. We can reduce this probability to a very small value if one of the nodes is allocated two time slots.

After node synchronization at deployment time, nodes continue to transmit during their own time slots. Every time a node hears a master packet with correct CRC, the node re-synchronizes clocks. This procedure keeps the network synchronized over time, and enables robustness to clock drifts.

3.4.2 Deployment and Ranging

We deploy the static sensor nodes easily without a careful measurement of their location–for example, tossing them off the side of the boat at convenient locations. These statically deployed sensor nodes communicate acoustically using the self-synchronizing TDMA algorithm (Section 3.4.1). During communication, ranges between pairs of nodes are obtained. There are three different methods to obtain the ranges with our acoustic modems. The first is to measure the round trip time of a message between a pair of nodes. This give us a ranges with an accuracy of ± 3 cm. The second method is to synchronize the clocks on the nodes and then use a schedule to determine when each node should send a message. The modems have temperature compensated oscillators with about one part per million drift which allows sub-meter ranging accuracy for about thirty minutes before the clocks need to be synchronized again. Finally, using a special broadcast message type the robot is able to obtain multiple ranges from round-trip messages in each slot. We observed experimentally between 1 and 4 ranges per TDMA slot. Thus the range to a node is received every 1 to 4 seconds.

Each node sequentially broadcasts its own ranges so that all nodes are able to build a table containing many of the inter-node ranges. After obtaining a sufficient number of inter-node ranges they self localize using an extended version of the robust static localization algorithm presented by Moore *et al.* [21]. This algorithm prevents nodes from being localized if there is a topological ambiguity so that errors are not propagated.

3.4.3 Tracking Algorithm Overview

The robot communicates with the network to obtain location information of the static nodes. After robot identifies the locations of the static nodes, it takes all of the TDMA slots, becoming the master of the whole network. The robot sends out a broadcast message which allows it to obtain ranges from multiple sources in quick succession. However, at times due to poor communication, no ranges are heard for time periods as long as tens of seconds. Even when receiving ranges once per second the robot can move significantly between range measurements. Thus, the tracking algorithm must explicitly account for and compensate for this motion.

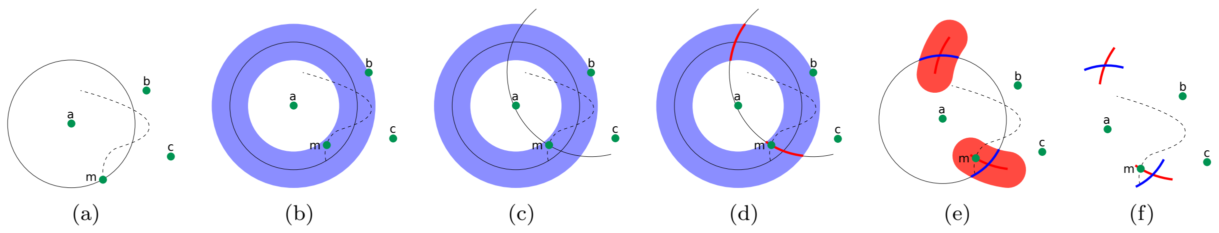


Figure 12: Example of the range-only localization algorithm.

Figure 12 shows the critical steps in the range-only tracking of mobile Node **m**. Node **m** is moving through a field of localized static nodes (Nodes **a**, **b**, **c**) along the trajectory indicated by the dotted line.

At time t Node **m** obtains a range to Node **a**. This allows Node **m** to localize itself to the circle indicated in Figure 12(a). At time $t+1$ Node **m** has moved along the trajectory as shown in Figure 12(b). It expands its localization estimation to the annulus in Figure 12(b). The size of the annulus is determined by the known maximum speed of the robot. Node **m** then enters the communication range of Node **b** and obtains a ranging to Node **b** (see Figure 12(c)). Next, Node **m** intersects the circle and annulus to obtain a localization region for time $t+1$ as indicated by the bold red arcs in Figure 12(d). This is the localization region. This region depends on the maximum speed of the robot and the specific ranges it hears.

Algorithm 3 Tracking Algorithm

```
 1: procedure LOCALIZE($A_1 \cdots A_t$)
 2:     $s \leftarrow$ max speed
 3:     $I_1 = A_1$                                    ▷ Initialize the first intersection region
 4:     for $k = 2$ to $t$ do
 5:         $\triangle t \leftarrow k - (k-1)$
 6:         $I_k =$ Grow$(I_{k-1}, s\triangle t) \cap A_k$       ▷ Create the new intersection region
 7:         for $j = k-1$ to 1 do                      ▷ Propagate measurements back
 8:             $\triangle t \leftarrow j - (j-1)$
 9:             $I_j =$ Grow$(I_{j+1}, s\triangle t) \cap A_j$
10:         end for
11:     end for
12: end procedure
```

The range taken at time $t+1$ can be used to improve tracking at time t as shown in Figure 12(e). The arcs from time $t+1$ are expanded to account for all locations the mobile node could have come from. This is intersected with the range taken at time t to obtain the refined location region illustrated by the bold blue arcs. Figure 12(f) shows the final result. Note that for times t and $t+1$ there are two possible location regions. This is because two range measurements do not provide sufficient information to fully constrain the system. Range measurements from other nodes will quickly eliminate this.

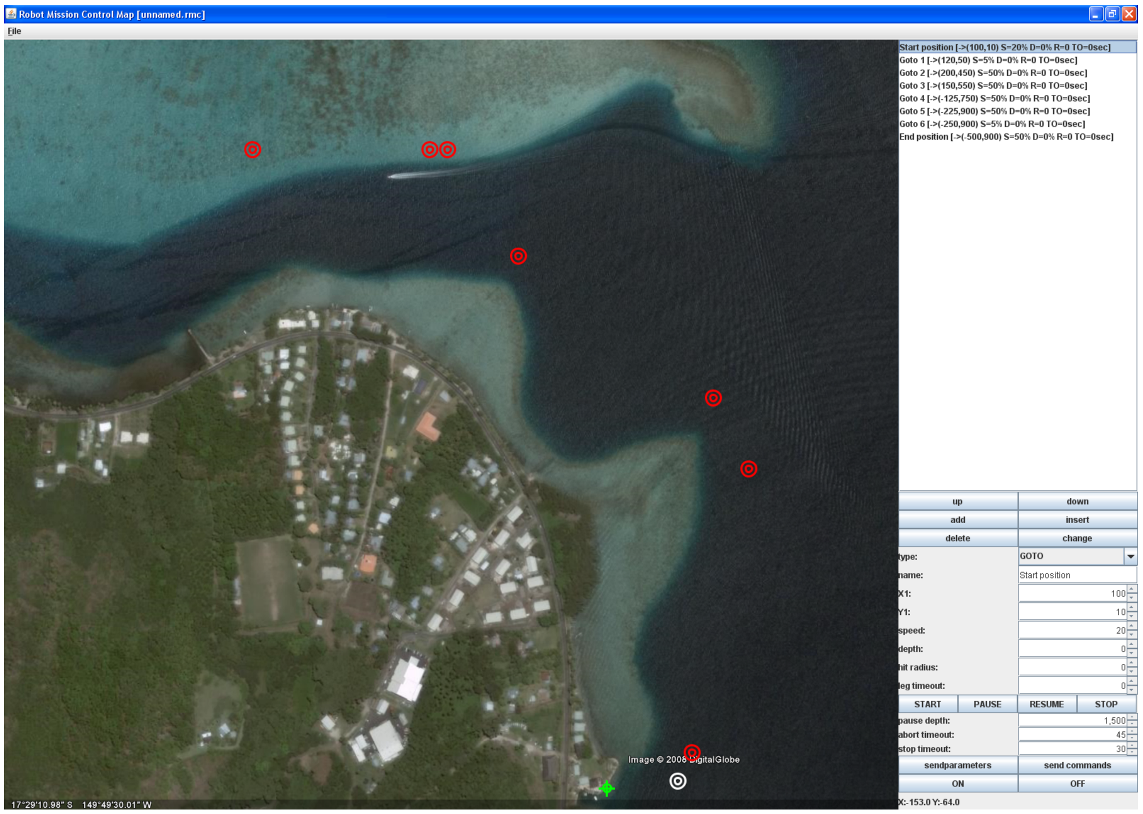
Pseudocode for the algorithm is shown in Algorithm 3 and follows the same idea as in Figure 12. At each step the goal is to compute an updated localized region for the robot. Each new region computed will be intersected with the grown version of the previous region and the information gained from the new region will be propagated backwards. "Growing" a region accounts for the possible motion of the robot during the time between range measurements (e.g. expanding the circle to an annulus).

The first step in Algorithm 3 (line 3), is to initialize the first intersection region to be the first region. Then we iterate through each successive region. The new region is intersected with the previous intersection region grown to account for any motion (line 6). Finally, the information gained from the new region is propagated back by successively intersecting each optimal region grown backwards with the previous region, as shown in line 9.

Algorithm 3 can be run online by omitting the outer loop (lines 4-6 and 11) and executing the inner loop whenever a new region/measurement is obtained.

We are able to prove that this algorithm produces the optimal localization regions. That is, the regions it finds are the smallest regions that must contain the true location of the mobile node. See [6] for details. This algorithm has a runtime of $O(n^3)$ where n is the number of input regions. However, we also show that the runtime of the online version can be reduced to $O(1)$ for actual implementations (where we can deal with very small errors).

We have implemented this algorithm in simulation and also on the robot and sensor network described in this paper.


Robot Mission Control Map [unnamed.rmc]
File
Image © 2008 DigitalGlobe
17°29'10.98" S 149°49'30.01" W
up
down
add
insert
delete
change
START
PAUSE
RESUME
STOP
sendparameters
send commands
ON
OFF

```
Start position [->(100,10) S=20% D=0% R=0 TO=0sec]
Goto 1 [->(120,50) S=5% D=0% R=0 TO=0sec]
Goto 2 [->(200,450) S=50% D=0% R=0 TO=0sec]
Goto 3 [->(150,550) S=50% D=0% R=0 TO=0sec]
Goto 4 [->(-125,750) S=50% D=0% R=0 TO=0sec]
Goto 5 [->(-225,900) S=50% D=0% R=0 TO=0sec]
Goto 6 [->(-250,900) S=5% D=0% R=0 TO=0sec]
End position [->(-500,900) S=50% D=0% R=0 TO=0sec]
```

type:	GOTO
name:	Start position
X1:	100
Y1:	10
speed:	20
depth:	0
hit radius:	0
leg timeout:	0
pause depth:	1,500
abort timeout:	45
stop timeout:	30

X:-153.0 Y:-64.0

Figure 13: Mission editor. The editor contains a map for visualization (left, taken from Google Earth), a list of mission legs (right top), and a control panel for mission editing and control (right bottom). Shown is a mission with 8 legs that commands the robot to drive from the dock to our test site. The green cross is the map origin. The green circle shows the current robot position. Red double circles show waypoints, the white circle is the currently active waypoint.

3.5 Mission Planning and Robot Interface

Amour V is equipped with a mission control module that allows it to autonomously travel on programmed paths and execute programmed dive profiles. For example, missions can be used to deploy Amour V from a dock instead of having to take a boat to the research site. Using missions Amour can autonomously travel to the measurement site, execute a preprogrammed dive profile to get sensor readings at desired locations, and finally return to the dock. Such missions can be extended to transport payloads from the dock to the measurement site and back.

Missions are created and edited by the user using a mission editor that displays the current mission on a map. During execution of a mission the user can follow the robot's location and state through the user interface of the mission editor if a connection with the robot exists. The robot sends its position and state information continuously over the both radio and acoustic links. The most recent information received by the user interface is used to display mission progress. Further, the user can instruct the robot to pause the execution of the mission, abort the mission, execute a different mission, or can be switched into manual operation.

A mission is represented as a path that the robot will follow during execution. A mission path consists of waypoints that are augmented with additional information like robot speed. Wait times can be inserted to get accurate sensor readings or to execute special tasks like placing or picking up a payload. The internal mission controller executes a mission path using the most accurate position information available at any time during the mission. This can include but is not limited to GPS data, acoustic position estimation with beacons, and dead reckoning. Additionally the robot can be programmed for arbitrary depth profiles which are followed closely using Amour V's pressure sensors for depth estimation.

Our software organizes missions as a sequence of mission legs with some additional global mission parameters. The global parameters are the origin of the mission coordinate system given in global GPS coordinates and a mission timeout after which the mission will be aborted whether the robot reached the final goal or not. Mission legs can represent waypoints if the robot is to follow a simple path, or other instructions such as a wait time for the robot to remain stationary. Waypoint information for each mission leg is stored as relative latitude and longitude position in reference to the mission coordinate system. Each mission leg further contains the desired travel speed, the depth at which the robot is to remain during the execution of this leg, and finally a distance which describes how accurate the goal coordinates must be approached. Every mission leg further contains a timeout after which the robot will abandon the current leg and try to execute the next mission leg. The timeouts are especially important for dive waypoints if the robot loses localization and thus cannot correctly identify when it has reached the goal. A mission leg can be configured such, that if it times out, the robot will automatically resurface.

4 Experiments

We conducted two types of experiments: in the pool and in the ocean. For pool tests we were able to mark the bottom of the pool and examine video data in order to extract the ground truth used to evaluate the results. During the ocean experiments the robot was driven close to the water surface so that a GPS antenna would gather information about the location of the robot we regard as ground truth. The relevant navigation data was logged and each experiment was also videotaped.

4.1 Basic motion, speed and attitude controller tests

The first series of experiments were done in MIT's Alumni Pool. The Pool has a maximum depth of 4 m and is 25 m long. The following experiments and tasks were evaluated in the pool:

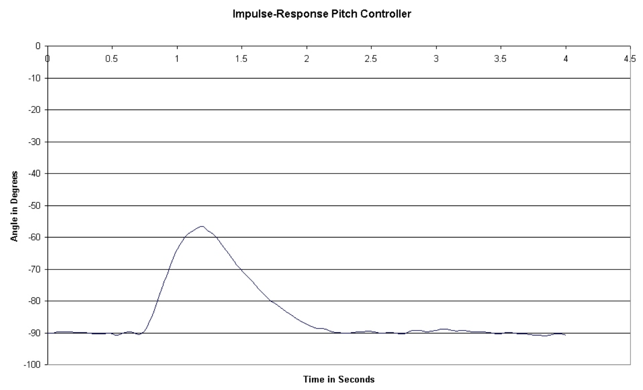
- Controller performance tests
- Maximum speed and power tests
- Remote controlled travel

- Autonomous drive test

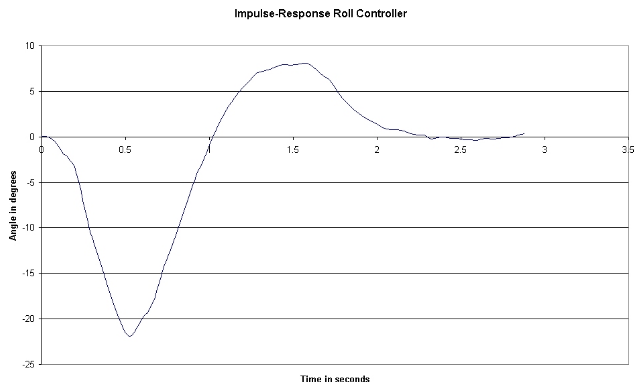
The controllers were tested by applying external forces to the robot while it was floating at a depth of 1.2 meters. The robot was manually tilted in all axes, one after another. Furthermore, it was forced to different depths. All input- and output-data was recorded using the control software's logging function. The experiment was also recorded by an underwater camera, whose time was synchronized to the timestamps in the log file. This was necessary to be able to assign the recorded data-sets to the respective movement. With the data gathered in these experiments impulse-responses of all controllers could be plotted.

The following figures show the results of the controller tests performed during the pool experiments. The robot was floating in horizontal orientation during the tests. Even though the control algorithms are the same in pitch, roll and yaw, there are differences in their behavior. This is due to a different thruster configuration in all axes. There are also different parameter sets, which were determined experimentally. Figures 14(b) and 15(b) show the robot's control response and demonstrated experimentally that the robot's controllers are stable and precise. More specifically:

- Given the current parameter set there is no overshoot in the pitch control (Figure 14(a)). The robot was pushed more than 30 degrees in this axis and it came back to the desired orientation within one second.
- The roll controller (Figure 14(b)) performed well, with a slight overshoot. The robot was moved about 22 degrees in this axis. It took almost two seconds until it came back to neutral after an overshoot of about 7 to 8 degrees. The gain of this controller's differential input should be increased for the next experiments in order to achieve better performance. Nevertheless, the current performance is still acceptable.
- The yaw controller (Figure 15(a)) works very well. There is minimal overshoot and it takes aproximately 1.5 seconds to compensate for an impulse of more than 45 degrees.
- The resolution of the pressure sensor is visible in the impulse-response plot of the depth controller in Figure 15(b). The pressure sensor (MPXA4250A), contains an internal ADC/DAC circuit with a resolution of about $5cm$ of water depth. The discrete nature of the output of this sensor was not mentioned in the datasheet. The robot was supposed to stay at a depth of 1.2 meters. It did come back to its original depth after it was moved up about 60 cm, but there was an overshoot. The current performance is acceptable, but can be improved by using a purely analog pressure sensor.

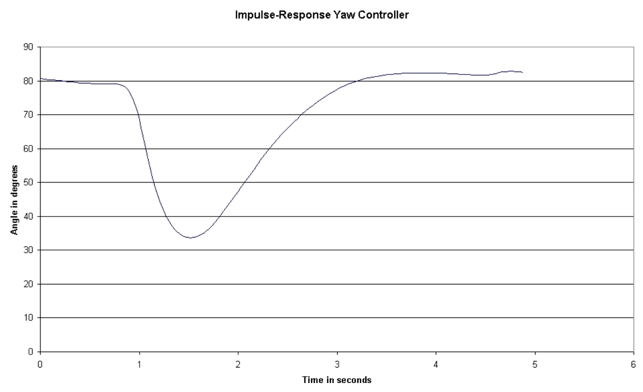


(a)

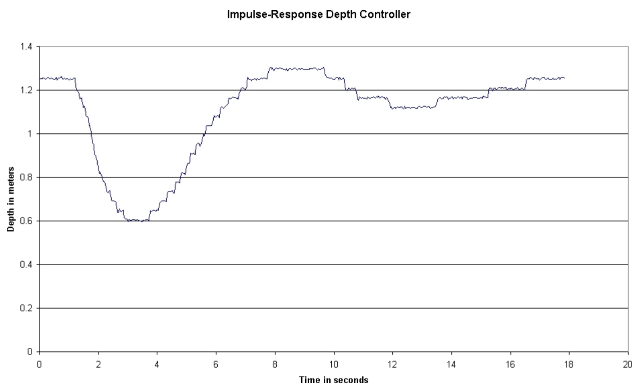


(b)

Figure 14: Impulse response of the (a) pitch and (b) roll controller



(a)



(b)

Figure 15: Impulse response of the (a) yaw controller and (b) depth controller

The 25m pool was used to test the maximum speed of the robot underwater. The robot was commanded to stay at $1m$ depth while driving full speed from one end to the other in horizontal and vertical configuration. A video camera was used to record the robot together with yard marks on the floor next to the pool to calculate the maximum speed.

The robot traveled a distance of about $20m$ in $16s$, resulting in an average speed of $v_{avrg} = 1.33m/s = 2.6knots$. The tests were repeated three times. In vertical configuration the speed was about 66% of the horizontal speed. The trade-off is maneuvrability.

We tested the robot's maneuvrability by commanding it to move along a square-shaped trajectory at $0.5m$ depth. 10 waypoints were used to describe the square and a turn of 90 at each corner (see Table 1). The robot motion was open loop and was observed from the surface. The robot performed the motion as expected.

Waypoint No.	Roll	Pitch	Yaw	Depth	Forward speed	Time [1/100s]
0	0	0	0	$0.5m$	0%	400
1	0	0	0	$0.5m$	20%	200
2	0	0	90	$0.5m$	0%	400
3	0	0	90	$0.5m$	20%	200
4	0	0	180	$0.5m$	0%	400
5	0	0	180	$0.5m$	20%	200
6	0	0	270	$0.5m$	0%	400
7	0	0	270	$0.5m$	20%	200
8	0	0	360	$0.5m$	0%	400
9	0	0	360	$0.5m$	20%	200

Table 1: Waypoints used for the autonomous drive test

4.2 Mission Control

In this section we describe a suite of experiments designed to test AMOUR V's ability to perform autonomous missions in natural ocean settings. We gave AMOUR V the goal to drive a total of 5 autonomous missions during the same experiment. Figure 16 shows the path traveled by the robot during one of these autonomous missions. The robot was instructed to drive to mission markers distributed at the corners of a 50m x 50m square. Each drive was represented by a mission leg. During the first of these mission legs the robot was instructed to dive to a depth of 5m and travel using dead reckoning. The rest of the mission was driven at surface level using GPS for localization. Because the robot's depth was too high to get GPS location and the underwater sensor network was not deployed along with the robot, the robot had to rely on dead reckoning. This is the reason why the robot did not estimate correctly the first waypoint, surfaced before reaching the marker for mission leg 1 (bottom right in Figure 16), and started to execute mission leg 2 (top right). The remaining three markers were reached to a desired error tolerance of 10m as planned. After reaching the last marker the mission controller switched off AMOUR V's motors and we were able to lift the robot into the boat.

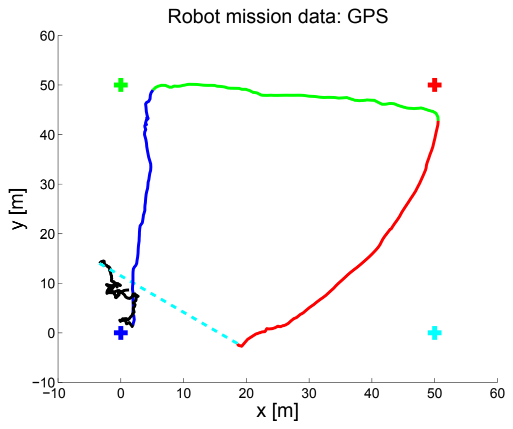


Figure 16: Autonomous mission path. The crosses denote the sides of a 50m by 50m square that the robot was told to drive. The dashed path was a dive at 5m depth while the other three sides were driven on the surface. Because the dive was executed with dead reckoning the robot surfaced early and started the next mission leg.

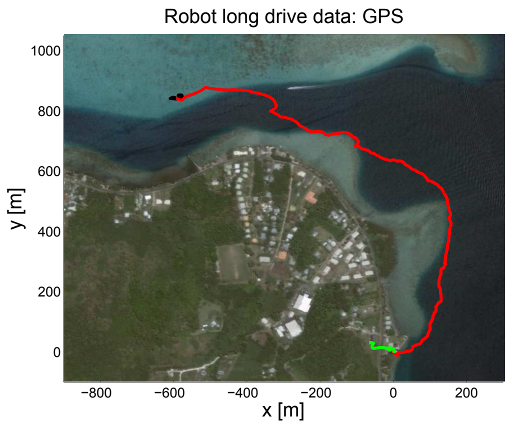


Figure 17: GPS path during endurance mission. The robot was driven by a human operator through a wireless link. The robot traveled at 75% of its maximum speed for a total of 45 minutes. The distance traveled was 1650 meters from the test site to the research station dock.

During the mission we monitored execution within the mission user interface when the robot's depth enabled a radio link. The robot drove a total of 200m for this particular mission (and each of the other four missions.) All five missions were completed within two hours including programming of the mission and the robot deployment. During one of the missions the robot was sent to a depth of 10m and resurface after a total of 50m traveled at that depth.

In a second suite of ocean experiments we tested the robot's endurance (e.g. battery life) and its ability to travel in realistic environments. We commanded the robot to travel in the open ocean and measured its performance over a 1.65km trajectory. The robot traveled at surface level to facilitate feedback and trajectory corrections transmitted by radio. The robot was observed and followed from the boat. The water current was 0.5 m/sec in the opposite direction of travel and the wave surf was 0.5m from lowest to highest surface point. The robot traveled at the surface in order to collect ground truth for its trajectory in the form of GPS waypoints (Figure 17). As a result it was exposed to the previously mentioned waves and water currents because of windy weather. The current was moving in the direction of the open ocean and out the bay (see Figure 17) and was thus working strongly against the robots intended direction of travel. The robot took 45 minutes to travel 1650 meters, which is equivalent to an absolute speed of 0.61 m/sec. Based on the thruster output, the robot should have traveled at 1 m/sec. The difference in speed coincides well with our estimate of water current speed. The battery was charged fully and used only for a few short system test runs before the actual experiment. After the endurance experiment the battery charge level was 50%. Thus with a fully charged battery the robot could have traveled 3.3 km against a water current of 0.5 m/sec and with a surf of 0.5m.

This experiment demonstrated the ability of the robot to travel long distances. The robot was able to travel on the surface in averse conditions.

4.3 Network and Tracking Experiments

The external localization and tracking system was built with the goal of easy deployment. The sensor nodes that comprise the underwater GPS/tracking system can be placed in water without careful consideration of their location. They automatically establish an underwater wireless network and localize themselves to create a system of coordinates using ranges obtained from the acoustic modems as described in Section 3.4

In this section we present data which characterizes the ranging capabilities and communication success rates for the acoustic modems. We then present the results of three different tracking experiments performed in Moorea, Singapore and Cambridge MA.

4.3.1 Ranging and Communication Experiments

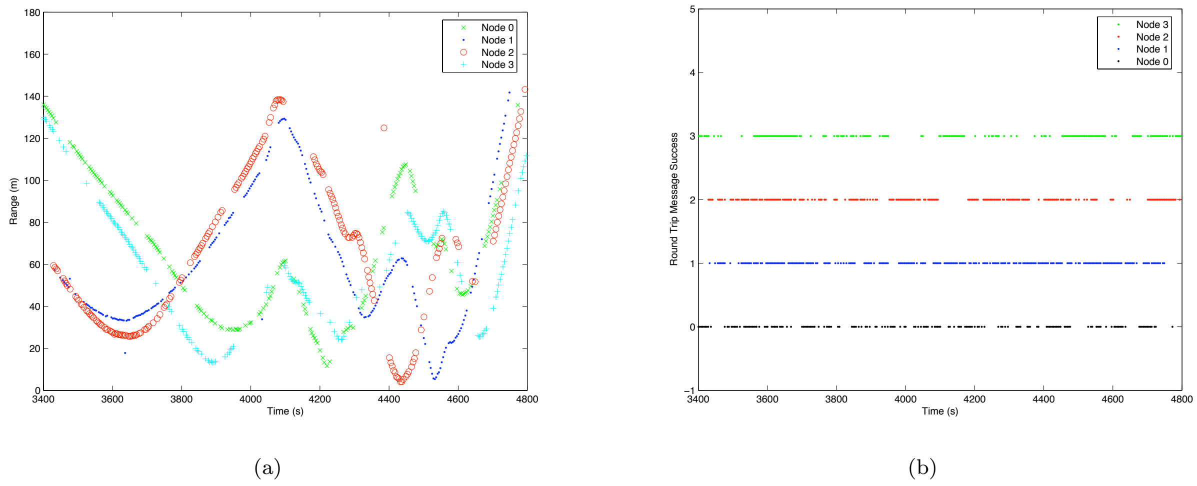


Figure 18: At left, are the ranges received by the robot while moving around in a network with four static nodes. At right, the successful round-trip messages from the robot to four different nodes is plotted.

Figure 18 shows the results of an experiment where four static nodes were deployed in Lake Otsego, NY. The depth where we deployed these nodes was approximately 10 meters deep and relatively clear and cool. The nodes self-localized and then we drove the robot through the network collecting ranging data to the nodes using the fast ranging mode where we receive ranges from approximately one node per second.

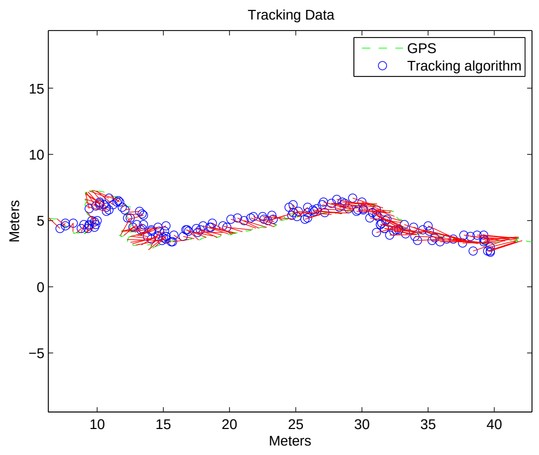
Figure 18(a) plots range data to the four nodes over a 20 minute selection of the hour long experiment. As can be seen from this plot, ranges are fairly continuous and there are very few outliers. Outliers are typically caused by receiving the multipath of a message instead of the direct path message. This causes an increase in the range measurement as the message took longer to arrive than typical. Additionally, there are a few measurements that are lower than expected. These can be explained by multipath messages from the previous time slot which arrive in the current time slot. This is one of the many challenges associated with underwater communication and ranging (since we use the round-trip time to compute ranges). We avoid this problem in most cases by having a sufficiently long enough guard time which allows most of the reflections to die down. These outliers can be eliminated using a filter.

This data also shows that we are able to successfully obtain ranges up to the maximum $140m$ range that we traveled. We have also successfully obtained ranges at up to $300m$, however, at this larger range the communication is less reliable.

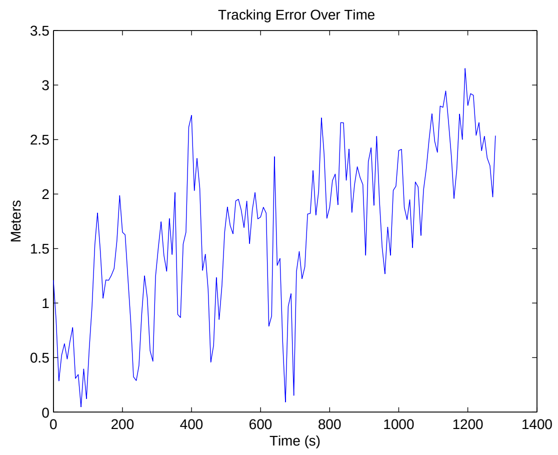
Figure 18(b) illustrates the round-trip message success rate for this same set of data. Over this experiment we found that about 59% of messages were successful. This is typical for most of our experimental sites except for the Charles River where we find a much lower message success rate.

Notice that there are periods where the robot is unable to obtain ranges for 10s of seconds from a particular node. Figure 18(a) shows that there is very little correlation between the distances at which these communication failures occur. For instance, the data for Node 2, shows that there is a period when no ranges are received when it is over $100m$ away, but there is a similar communication gap at $20m$, later in the experiment. Other times the robot is able to obtain most all messages to Node 2 at over $100m$. This indicates that the communications success rate is not merely a function of range, rather it is highly depending on the particular configuration of the nodes and the water conditions.

4.3.2 Tracking Experiments



(a)

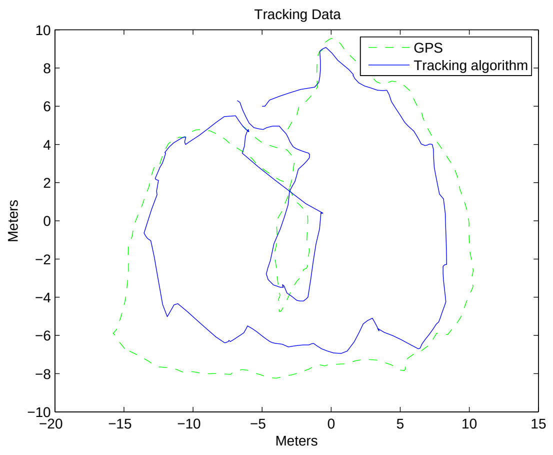


(b)

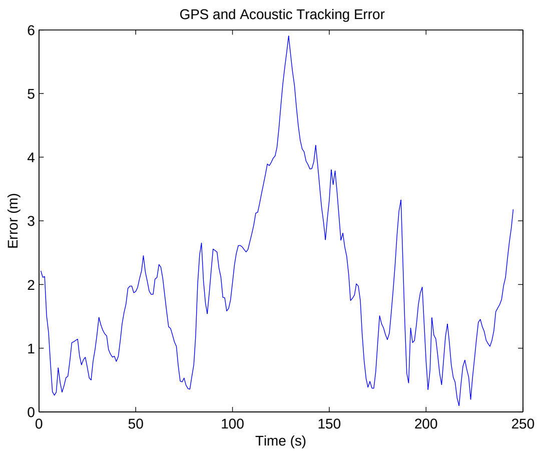
Figure 19: Results of the tracking algorithm in Singapore. At left, the comparison between the gps (dashed) and tracking algorithm (circles) with lines drawn between corresponding points. At right, the error between the readings in meters.

We have conducted robot tracking and localization experiments using AMOUR V and six underwater sensor nodes at three different locations: in an open bay in Moorea where the water depth was on the average 5m, off a dock in Singapore where the water depth was on the average 5m, and in the Charles river in Cambridge MA. In each case the robot traveled on the surface of the water in order to collect GPS position information which is used as ground truth when comparing to the position information computed by the underwater sensor network. In each case the static nodes were deployed in an approximate grid configuration and covered an area on the order of $30m \times 30m$. Each node was anchored in water at a depth of approximately $2m$. The node was deployed by manually throwing it overboard at the approximate desired location. The exact location of the node was established by the underwater sensor network using the underwater location algorithm. In each experiment, the robot traveled in the area covered by the sensor network, along different trajectories. We collected tracking data for the robot for approximately 2 hours in Moorea, 7 hours in Singapore, and 2 hours in the Charles river. We have also performed similar experiments at these locations. Subsets of the collected data are presented for each of the three locations.

Figure 19 shows a data segment from the localization and tracking experiment conducted in Singapore in May 2009. In this experiment we deployed six AquaNodes in static locations off a dock. The static sensor nodes self localized in an area of approximately 30 by 30 meters. The self-localization took approximately 30 minutes. The system requires this time to obtain enough range information to statically locate the static nodes and establish a system of coordinates.



(a)



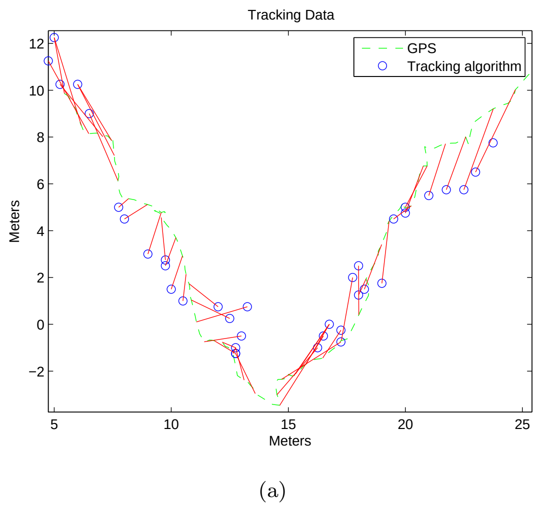
(b)

Figure 20: At left, the GPS data compared to the path recovered by the acoustic tracking algorithm in Moorea. At right, the error between the readings in meters.

The robot moved through the field of static nodes dynamically computing its own location based on the range information received from the sensor nodes. Figure 19(a) shows a detailed comparison of the GPS readings obtained at the surface and the acoustic tracking algorithm over a segment of the mission when the robot was at the surface. Circles are drawn to indicate the acoustic tracking location and red line segments are drawn to show the corresponding GPS position. Figure 19(b) shows the difference between the GPS and tracking algorithm over time. The maximum error is about 3 meters an average error of 1.67 meters.

Figure 20 shows sample results of experiments with localization and tracking performed in August 2008 at the UC Berkeley Gump Marine Research Station in Moorea, French Polynesia. This experiment was conducted with the robot at the surface, so it was able to obtain GPS positions as well as tracking data from the acoustic network. The first thirty minutes of the experiment were used for the sensor nodes to perform self-localization. The nodes for this experiment were deployed in a 20 by 30 meter area. The robot then moved through the field collecting ranges to the sensor nodes. The blue lines shows the results of our acoustic tracking algorithm and the green line shows the GPS ground truth. The median error in this experiment was 1.7 meters which is well within the GPS error.

Figure 21 shows an experiment performed in the Charles River. In this experiment we deployed 5 static nodes for approximately one hour to collect static range information. The nodes were deployed in an area of approximately 20 by 30 meters. After this time only four of the nodes were able to localize themselves due to very poor acoustic communications in the river environment. We suspect that the poor communication is



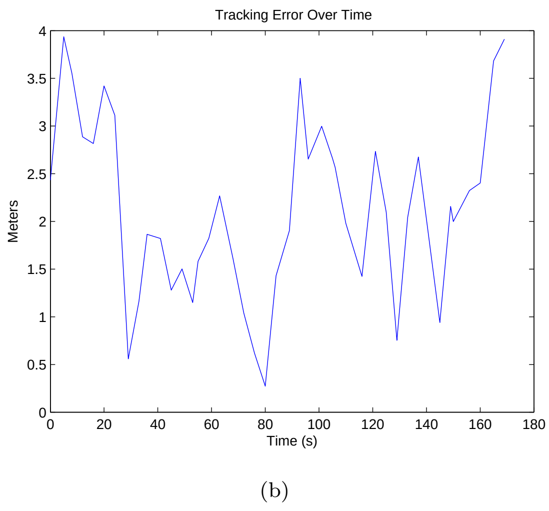


Figure 21: At left, the GPS data compared to the path recovered by the acoustic tracking algorithm in Charles River. At right, the error between the readings in meters.

due to the relatively shallow water where we deployed the network (3 meters) as well as reflections from the long pier we were deploying some of the nodes from. Due to the poor acoustic communications in the river the mobile node was only able to obtain location information sporadically during the experiment. Figure 21 shows three minutes of relatively good tracking results. However, there were often periods of more than a minute where only a few ranges were received.

We have tested our localization and tracking system in a variety of different water conditions. We found the most difficult environment to be the shallow river basin due to the poor acoustic communications in this environment. However, when we were able to obtain regular ranges from the acoustic modem we have shown that we can compute the location of the robot to within a few meters of the GPS ground truth. This error is well within the error of the GPS receiver.

4.4 Buoyancy and Balance

Several experiments were done to evaluate the buoyancy and balance control. The buoyancy experiment examines the aggregate thruster output as the robot's weight is changed by adding or removing a weight of 950g. Figure 22 shows the results. At time $t = 0$ the robot hovers at constant depth in vertical configuration. The thrusters oscillate around 0% output. The robot's depth is maintained at 0.5m $(+/-2\text{cm})$ below the water surface. The depth controller output oscillations are due to the latency of the sensorless motor controllers while changing direction (we note that the larger oscillations are due to human touch in preparation for adding a weight). At time $t = 100$s a weight of 950g is added to the robot. The depth controller oscillates till time $t = 110$s in response to this event. Between times $t = 110$s and $t = 170$s the depth controller is stable at 40% output to compensate for the additional weight. We note that the oscillations are smaller in this phase since the controller does not need to change the direction of rotation as in the neutrally buoyant case. At time $t = 170$s the buoyancy engine is activated. Between $t = 170$s and $t = 185$s the buoyancy engine effects work to compensate for the additional weight of the robot. At time $t = 185$s the robot is again neutrally buoyant and the depth controller operates around 0%. At time $t = 230$s the robot releases the attached weight and the depth controller starts compensating for the positive buoyancy of the robot. At time $t = 250$s the buoyancy engine is activated again to bring the robot back to neutral buoyancy which happens at time $t = 265$s. We have repeated the buoyancy control experiment 10 times with similar results.

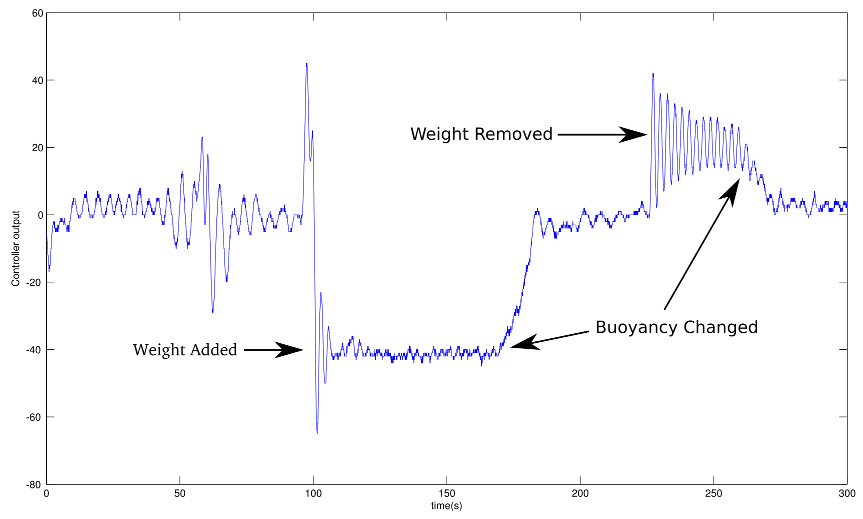


Figure 22: Buoyancy experiment: thruster controller output (in percentage) over time as we vary the robot's weight.

The average time required to compensate for the robot's weight change is 15s.

The next group of experiments illustrates the effect of moving the battery inside robot on the pitch controller. Figure 23 shows the data. The robot starts in horizontal configuration with a 950g weight attached to its bottom. The battery is at its bottom position in the robot's cylindrical enclosure. The thrusters work at -40% to compensate for the weight. At time $t = 10$s the battery begins moving toward its top-most position. It takes 25s to complete the move. We observe the controller output at 10% (in the other direction). This means that the robot's battery can compensate for more than the 950g attached to it. The movement is repeated back and forth.

A third suite of experiments concerns balance control. Figure 24 shows the data. In these experiments the robot starts by hovering in horizontal configuration with the battery placed at the bottom of the robot's cylindrical body. The thrusters keep the robot hovering in horizontal stance. At time $t = 20$s the battery is moved to its mid-point in the body of the robot and subsequently we observe the effect on the pitch controller's output. At time $t = 45$s a weight of 950g is added manually at the rear end of the robot and again we observe the effect on the pitch controller's output. At time $t = 67$s the robot starts moving the battery autonomously to adjust the robot's balance. At time $t = 85$s we observe that the robot has successfully achieved neutral balance and subsequently the pitch controller oscillates around 0. The balance control experiment was repeated 10 times with similar performance. The average time to achieve neutral balance was 20s.

5 Conclusions

In this paper we described a light-weight modular robot capable of hovering and navigation. Modularity is represented in this robot at several levels. The robot can support an arbitrary number of thrusters (we used it with four and five thruster configurations). At least four thrusters are needed to control translations, rotations, and hovering. Adding more thrusters to the robot gives the robot a higher maximum speed.

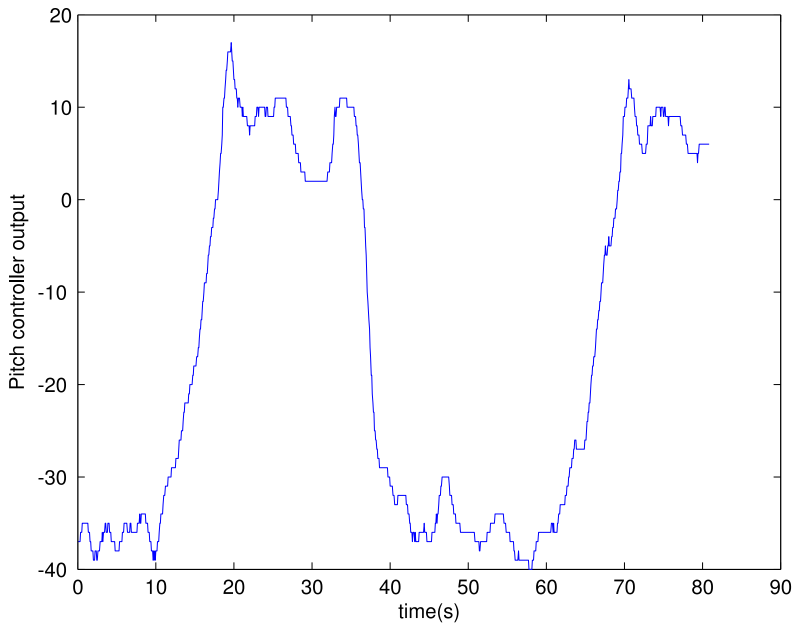


Figure 23: Battery experiment: thruster controller output (in percentage) over time as we change the robot's balance by moving the battery's position.

Additional modules include modules for communications, computation (e.g. high-level mission planning,) and perception (e.g. camera sensing, sonar sensing, and sensing for water chemistry.) The robot also includes optional docking modules that enable it to pick up a payload, and a balance and buoyancy control module that enable the robot to adjust its static buoyancy and balance to save energy when its weight changes by picking up or dropping a payload during a mission. We described the hardware used to achieve this kind of adaptive control and the supporting algorithms. We showed experimentally that this system can reliably save energy with buoyancy and balance control. We believe that simultaneous buoyancy and balance control will enable underwater robots to carry out pick and place operations efficiently and plan to demonstrate this point in a new suite of experiments to deploy and reposition an underwater sensor network.

Acknowledgments

The ocean experiments described in this paper were done in Singapore, at the Gump Marine Biology Station in Moorea, and in the Charles River. We thank the staff at Gump, and especially Frank Murphy for all their support. We thank Randy Davis for the use of his pool, Tim Beaulieu for supporting our pool work, and Jody Mello for the use of MIT facilities. This work was supported in part by DSO, NSF grants IIS-0426838, EFRI-0735953 and the MURI Antidote project. We are grateful for this support.

References

[1] G.I. Allen, G. Sulzberger, J.T. Bono, J.S. Pray, and T.R. Clem. Initial evaluation of the new real-time tracking gradiometer designed for small unmanned underwater vehicles. In *OCEANS, 2005. Proceedings of MTS/IEEE*, pages 1956–1962 Vol. 3, 2005.

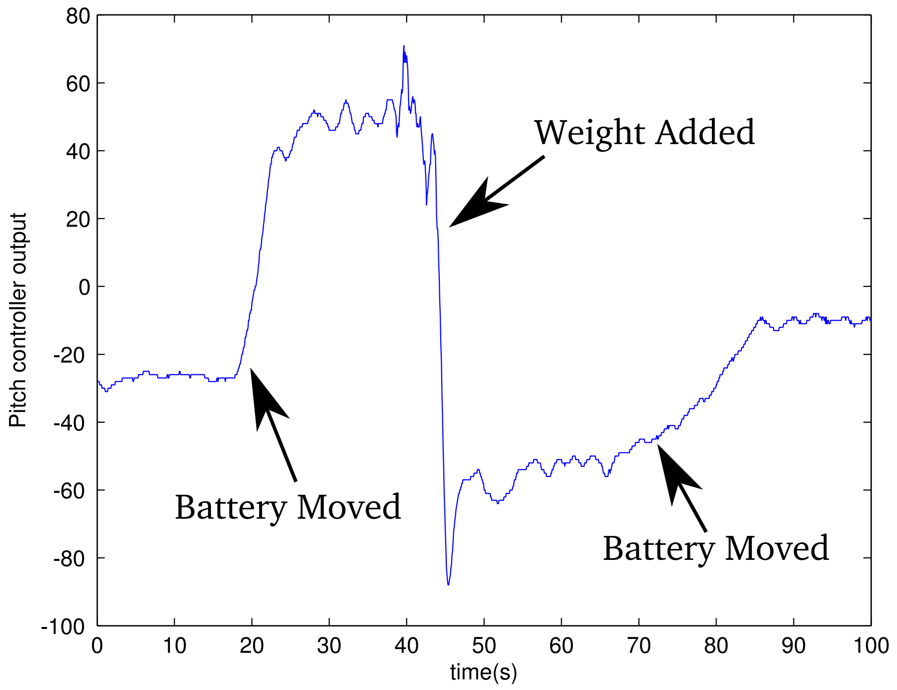


Figure 24: Balance experiment: thruster controller output (in percentage) over time as we change the robot's balance by adding weight.

[2] Gianluca Antonelli, Thor I. Fossen, and Dana R. Yoerger. *Underwater Robotics*, pages 987–1008. 2008.

[3] H.T. Choi, A. Hanai, S.K. 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*, volume 1, pages 836–841 vol.1, 2003.

[4] Peter Corke, Carrick Detwiler, Matthew Dunbabin, Michael Hamilton, Daniela Rus, and Iuliu Vasilescu. Experiments with underwater robot localization and tracking. In *ICRA07 - International Conference on Robotics and Automation*, April 10-14 2007.

[5] D.M. Crimmins, C.T. Patty, M.A. Beliard, J. Baker, J.C. Jalbert, R.J. Komerska, S.G. Chappell, and D.R. Blidberg. Long-endurance test results of the solar-powered auv system. In *OCEANS 2006*, pages 1–5, 2006.

[6] C. Detweiler, J. Leonard, D. Rus, and S. Teller. Passive mobile robot localization within a fixed beacon field. In *Proceedings of the 2006 International Workshop on Algorithmic Foundations of Robotics*, New York, aug 2006.

[7] C. Detweiler, S. Sosnowski, I. Vasilescu, and D. Rus. Saving energy with buoyancy and balance control for underwater robots with dynamic payloads. In *Proceedings of the 11th International Symposium on Experimental Robotics*, Athens, Greece, July 2008.

[8] C. Detweiller, I. Vasilescu, and D. Rus. An underwater sensor network with dual communications, sensing, and mobility. In *OCEANS 2007 - Europe*, pages 1–6, 2007.

[9] Gregory Dudek, Philippe Giguere, Chris Prahacs, Shane Saunderson, Junaed Sattar, Luz-Abril Torres-Mendez, Michael Jenkin, Andrew German, Andrew Hogue, Arlene Ripsman, Jim Zacher, Evangelos Milios, Hui Liu, Pifu Zhang, Marti Buehler, and Christina Georgiades. Aqua: An amphibious autonomous robot. *Computer*, 40(1):46–53, 2007.

[10] 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*, pages 2105–2110, 2005.

[11] M. Dunbabin, I. Vasilescu, P. Corke, and D. Rus. Cooperative navigation experiments with two autonomous underwater robots. In *ISER06 - International Symposium on Experimental Robotics*, 2006.

[12] M. Dunbabin, I. Vasilescu, P. Corke, and D. Rus. Data muling over underwater wireless sensor networks using an autonomous underwater vehicle. In *Proceedings of the 2006 International Conference on Robotics and Automation*, Orlando, FL, May 2006.

[13] C.C. Eriksen, T.J. Osse, R.D. Light, T. Wen, T.W. Lehman, P.L. Sabin, J.W. Ballard, and A.M. Chiodi. Seaglider: a long-range autonomous underwater vehicle for oceanographic research. *Oceanic Engineering, IEEE Journal of*, 26(4):424–436, Oct 2001.

[14] G. Bishop G. Welch. An introduction to the kalman filter,university of north carolina, 2006.

[15] D. Gurdan. Design and motion control of an autonomous underwater vehicle. Diploma thesis, TUM / MIT, Munich, jan 2007.

[16] Clayton Kunz, Chris Murphy, Richard Camilli, Hanumant Singh, John Bailey, Ryan Eustice, Michael Jakuba, Ko ichi Nakamura, Chris Roman, Taichi Sato, Robert A. Sohn, and Claire Willis. Deep sea underwater robotic exploration in the ice-covered arctic ocean with auvs. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3654–3660, 2008.

[17] J. L. Laine, S. A. Nichols, D. K. Novick, P. D. O Malley, D. Copeland, and M. C. Nechyba. Subjugator: a highly maneuverable, intelligent underwater vehicle. Technical report, Machine Intelligence Laboratory, University of Florida, Gainesville, FL, 1999.

[18] S. Licht, V. Polidoro, M. Flores, F.S. Hover, and M.S. Triantafyllou. Design and projected performance of a flapping foil auv. *Oceanic Engineering, IEEE Journal of*, 29(3):786–794, 2004.

[19] S.C. Martin, L.L. Whitcomb, D. Yoerger, and H. Singh. A mission controller for high level control of autonomous and semi-autonomous underwater vehicles. In *OCEANS 2006*, pages 1–6, 2006.

[20] D. McFarland, I. Gilhespy, and Honary E. Divebot: A diving robot with a whale-like buoyancy mechanism. *Robotica (2003)*, 21(04), aug 2003.

[21] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *Proceedings of the 2nd ACM SenSys*, pages 50–61, Baltimore, MD, nov 2004.

[22] P. Newman and J. Leonard. Pure range-only sub-sea slam. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1921–1926, Taiwan, 2003.

[23] D. K. Novick, R. Pitzer, B. Wilkers, C. D. Crane, E. de la Iglesia, and K. L. Doty. The development of a highly maneuverable underwater vehicle. In *Robotics 98: The 3rd International Conference and Exposition/Demonstration on Robotics for Challenging Environments*, pages 168–173, Albuquerque, NM, apr 1998.

[24] Edwin Olson, John Leonard, and Seth Teller. Robust range-only beacon localization. In *Proceedings of Autonomous Underwater Vehicles*, pages 66–75, 2004.

[25] Mehul Sangekar, Mandar Chitre, and Koay Teong Beng. Hardware architecture for a modular autonomous underwater vehicle starfish. In *OCEANS*, Quebec City, Canada, 2008.

[26] J. Sherman, R.E. Davis, W.B. Owens, and J. Valdes. The autonomous underwater glider 'spray.'. In *IEEE Oceanic Engineering*, number 26, pages 437–446, 2001.

[27] Hanumant Singh, Ali Can, Ryan Eustice, Steve Lerner, Neil McPhee, Oscar Pizarro, and Chris Roman. Seabed auv offers new platform for high-resolution imaging. *Eos*, 85(31):TRANSACTIONS AMERICAN GEOPHYSICAL UNION, August 2004.

[28] S. Sosnowski. Design and buoyancy control of an autonomous underwater vehicle. Diploma thesis, TUM / MIT, Munich, oct 2007.

[29] R.P. Stokey, A. Roup, C. von Alt, B. Allen, N. Forrester, T. Austin, R. Goldsborough, M. Purcell, F. Jaffre, G. Packard, and A. Kukulya. Development of the remus 600 autonomous underwater vehicle. In *OCEANS, 2005. Proceedings of MTS/IEEE*, pages 1301–1304 Vol. 2, 2005.

[30] J. Stumpf. Design and motion planning of an autonomous underwater vehicle. Diploma thesis, TUM / MIT, Munich, jan 2007.

[31] J. Vaganay, J. G. Bellingham, and J. J. Leonard. Comparison of fix computation and filtering for autonomous acoustic navigation, 1998.

[32] 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*, pages 1–6, 2006.

[33] I. Vasilescu, K. Kotay, D. Rus, M. Dundabin, and P. Corke. Data collection, storage, and retrieval with an underwater sensor network. In *Proceedings of the 3rd ACM SenSys*, San Diego, CA, nov 2005.

[34] I. Vasilescu, P. Varshavskaya, K. Kotay, and D. Rus. Autonomous modular optical underwater robot (amour): design, prototype and feasibility study. In *Proceedings of the 2005 International Conference on Robotics and Automation*, Barcelona, apr 2005.

[35] Iuliu Vasilescu, Carrick Detweiler, and Daniela Rus. Aquanodes: an underwater sensor network. In *Proceedings of the second workshop on Underwater networks*, pages 85–88, Montreal, Quebec, Canada, 2007. ACM.

[36] D. R. Yoerger, A. M. Bradley, S. C. Martin, and L. L. Whitcomb. The sentry autonomous underwater vehicle: Field trial results and future capabilities. *AGU Fall Meeting Abstracts*, 33:1674, December 2006.

[37] J. Yuh and M. West. Underwater robotics. *International Journal of Advanced Robotics*, 15(5):609–639, 2001.