Backseat Control of SandShark AUV using ROS on RaspberryPi*

John E. Naglak[1], Brian R. Page[1], and Nina Mahmoudian[2]

***Abstract*— This paper presents a backseat controller suitable for Autonomous Underwater Vehicles (AUVs) which promotes quick development turn-around of novel control methods, especially for missions which require onboard re-planning. This backseat controller communicates low-level control instructions to a vehicle-specific frontseat controller. Small, power efficient components comprise the backseat controller hardware, especially a Raspberry Pi computer. The backseat controller runs Robot Operating System (ROS), a middleware that facilitates implementation of novel control algorithms and use of OpenCV, an extensive collection of computer vision libraries. ROS enables the underwater community to leverage development efforts in the ground and aerial domains to accelerate progress. The backseat controller has been mated to General Dynamics Bluefin SandShark AUV. The platform has been validated in a controlled environment and is now being deployed in Lake Superior, MI, USA.**

***Index Terms*— Marine Robotics; Software, Middleware and Programming Environments; AUV; ROS; Backseat Control**

I. Introduction and Motivation

One obstacle to the development of tools to survey and explore the ocean is the difficulty of control development on large, high overhead, oceanographic vehicles. A small, modular, Autonomous Underwater Vehicle (AUV) is a good platform for controls development which enables tasks lacking a known or pre-planned trajectory. There are many applications for onboard trajectory calculation, including infrastructure inspection, mine countermeasures, and wildlife monitoring. Most AUVs are designed for area survey missions, and are equipped with a “frontseat” controller which handles low-level hardware control and failsafe operation, but are limited to simple preplanned trajectories with dead reckoning for localization while submerged. Work is ongoing to implement advanced localization methods on small AUVs [1], [2]. Development of a powerful, general purpose, “backseat” controller which calculates on-board trajectory updates based on sensor inputs and communicates instructions to the vehicles frontseat controller is a prerequisite to controls research [3], [4].

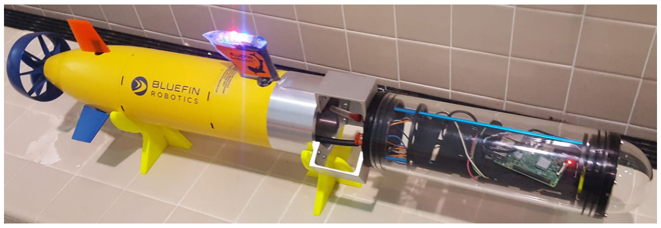


Fig. 1: The Bluefin SandShark AUV (yellow) with custom payload (clear). The backseat controller is housed inside of the payload pressure vessel and connected to the frontseat controller over an ethernet connection.

The Nonlinear and Autonomous Systems Lab (NASLab) at Michigan Technological University has developed and integrated a backseat controller for the General Dynamics Bluefin SandShark AUV. Although other groups have pursued similar development using the Mission Oriented Operating Suite (MOOS) [5], [6], integration of the Robot Operating System (ROS) as middleware is an important alternative which promotes transference of code and skill-set across domains (air, land, and water). The assembled SandShark

*This work is partially supported by ONR N00014-15-1-2599 and NSF 1453886.

[1]John Naglak and Brian R. Page are Graduate Research Assistants with the Department of Mechanical Engineering-Engineering Mechanics, Michigan Technological University, Houghton, MI, USA `jenaglak@mtu.edu, bpage@mtu.edu`

[2]Dr. Nina Mahmoudian is the Lou and Herbert Wacker Associate Professor in Autonomous Mobile Systems with the Department of Mechanical Engineering-Engineering Mechanics, Michigan Technological University, Houghton, MI, USA `ninam@mtu.edu`

AUV with custom payload is presented in Fig. 1. Due to the standard payload interface defined by Bluefin [7], this frontseat/backseat architecture will support all updated Bluefin Vehicles. The purpose of this paper is to detail the components and operation of this backseat controller, describe the interface with SandShark's frontseat, and validate its operational use.

II. Development and Implementation

Development of a backseat controller that enables ROS integration onto Bluefin vehicles was completed both in hardware and software. The backseat controller builds upon work completed in the open-source community and allows the SandShark to utilize powerful tools created for robotics in the ground and aerial domains.

A. Hardware Components

The SandShark is a small, modular AUV designed as an accessible platform for development and discovery. SandShark has a single aft propeller and a three-fin control surface configuration. The frontseat controller, also housed in the aft section, has access to GPS, Attitude Heading and Reference System (AHRS), altimeter, and pressure/temperature sensor. The vehicle is designed to be expandable through a modular design where the entire front half of the vehicle is reserved for user payloads. This user payload must be relatively small, as the vehicle is not powerful enough to accommodate large additional drag and its control surfaces are small. Also, the vehicle as shipped is beach-deployable. A large payload would preclude this. SandShark is capable of providing power and communication through a payload interface that follows the NMEA standard. All Bluefin vehicles utilize custom message types specified as part of the Huxley controller [7].

Located in the custom payload is a Raspberry Pi running Ubuntu MATE. The Raspberry Pi is a small, single board computer which includes a 64-bit quad core processor running at 1.2 Ghz, 1 GB of RAM, and integrated wireless LAN. This platform provides sufficient computational power for real-time data processing and robotic controls, while occupying a small volume in the payload. The peak power consumption of the Raspberry Pi is 2.4 watts at max CPU load, this is notable as an order-of-magnitude less than that of the common mini-ITX type computer or Odroid XU4. The hardware layout of payload is shown in Fig. 2. One useful feature of a traditional computer that the Rasberry Pi lacks is a Real Time Clock (RTC). This presented some difficulty in practice, which was resolved by sourcing an external RTC. The Pi was modified to support an external WiFi antenna. Attaching a high-gain antenna substantially improved communication with the Pi in open water testing, while still utilizing the built in WiFi hardware. Power for the payload is provided by SandShark's main battery. The SandShark Graphical User Interface (GUI) software provides user control of the payload power. A buck regulator provides 5v to the standard power connector for the Raspberry Pi. Peripherals within the payload are powered by the USB ports, including a Blue Robotics Low-Light HD camera.

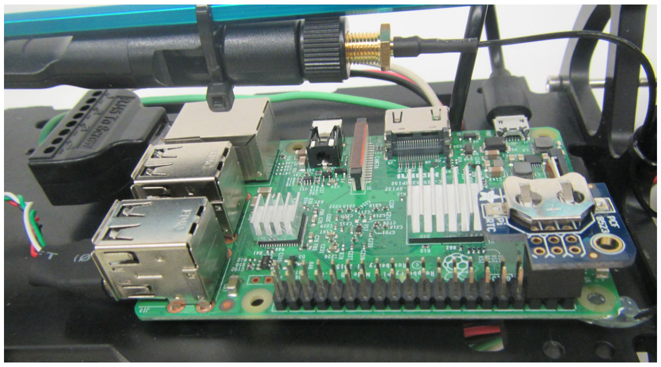


Fig. 2: A small user payload contains the backseat controller and supporting hardware.

B. Software Configuration

ROS and OpenCV have been implemented onto our custom payload to leverage the tools created by the broader robotics community for accelerated development. ROS is an open-source publisher-subscriber middleware which eases implementation of many of the data handling and computational tasks operating on an autonomous vehicle, including AUV's [8]–[10]. ROS also incorporates powerful logging and testing tools, as well as vast software package support for robotic algorithms.

We have installed ROS on the Raspberry Pi and written drivers which initialize communication with SandShark according to Bluefin requirements

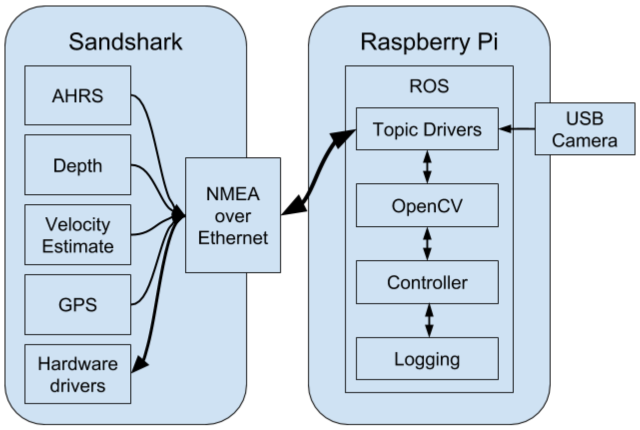


Fig. 3: SandShark provides hardware interface, power, and sensor data. The backseat controller runs vision processing, control, and datalogging using ROS.

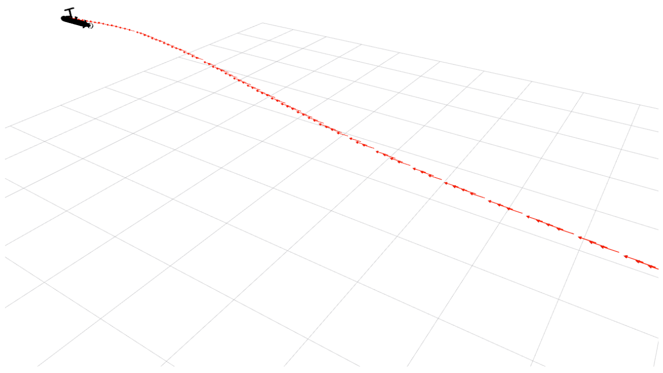


Fig. 4: RVIZ can depict AUV trajectories, in this case an approach to an optical beacon. The red arrows indicate the pose of the robot at each odometry update. Grid squares are 1 m^2.

and stream data from the frontseat to ROS Topics. An essential component of the drivers is the ability to pass low-level control commands back to SandShark. There are a number of permissible configurations for these commands, which include angular setpoints for rudder and elevator, as well propeller RPM, but can be changed to select some of the vehicles control laws, such as a heading or vehicle pitch setpoint. This communication layout is represented in Fig. 3. The core of the communication driver with SandShark is a python socket over ethernet, which passes NMEA type messages in plain text. We decode these messages and publish the data to a combination of ROS-standard and also custom topics.

Some critical ROS packages should be mentioned for this implementation. The robot_upstart package provides stable initialization of the backseat controller on boot. The usb_cam drivers work with the Blue Robotics Low-Light HD camera, although not all of the cameras firmware-supported configurations are accessible via this driver. UDEV rules were implemented to ensure that the camera initializes deterministically on boot with correct user permissions. Cv_bridge package handles the conversion between ROS image topics and OpenCv iplimage stuct. A family of packages have been included to ensure compatibility with the RVIZ environment. Robot_state_publisher, joint_state_publisher, and urdf are utilized with a simple model to support a nav_msgs/Odometry message. This message is populated with data from the front seat controller and used in RVIZ to visualize on-board trajectory estimation in post processing, Fig. 4.

OpenCV is an extensive collection of open source computer vision libraries. The Raspberry Pi runs OpenCV within ROS for processing and analysis of the video stream from a USB camera. The ROS camera driver provides a 640x480 px YUYV image streaming at 15 fps. The image is then pushed through the cv_bridge converter to be available for OpenCV processing. We also publish a visualization of the image processing result on an additional ROS topic, which allows us to debug image processing in real-time on the bench, or bag the result and review it after the vehicle surfaces.Periodically, especially during open-water testing, the raw camera output is bagged to refine the image processing method offline.

III. Validation and Application

Initial testing and development of the frontseat-backseat architecture has been completed in the pool environment at Michigan Tech. Throughout the past year the controller has operated for over 60 hours of active in-water testing. At this time, the vehicle is operating in the open-water environment. To validate the choice of Raspberry Pi as backseat controller the power, processor, and memory loads were recorded during typical operations. Power consumption of the payload is 0.26 A at 27.5 Vdc, under full computational load. Average CPU load across all cores is around 50.4% and average RAM consumption is 33.6%.

Aided by the ROS installation and low-level interface with the Huxley software, the frontseat-backseat architecture on the SandShark AUV can be configured to complete a wide variety of missions. Currently the vehicle is configured to perform target approach missions using visual feedback when available during the approach maneuver. To accomplish this, the SandShark frontseat controller operates in an out-of-the-box waypoint navigation mode until the optical markers are available. The backseat then takes over and calculates relative position and attitude of the optical beacon while driving the vehicle towards the beacon using a custom control strategy. The handoff between frontseat and backseat enables the vehicle to operate as expected out of the box during the majority of mission time with mission modifications occurring when required by the backseat. Fig. 5 shows the AUV approaching a beacon in Lake Superior under control from the algorithms running on the backseat. The associated approach vision processing result, recorded in a ROS bag file, can be seen in Fig. 6. An example of rudder and pitch control commands throughout this approach sequence are in Fig. 7.

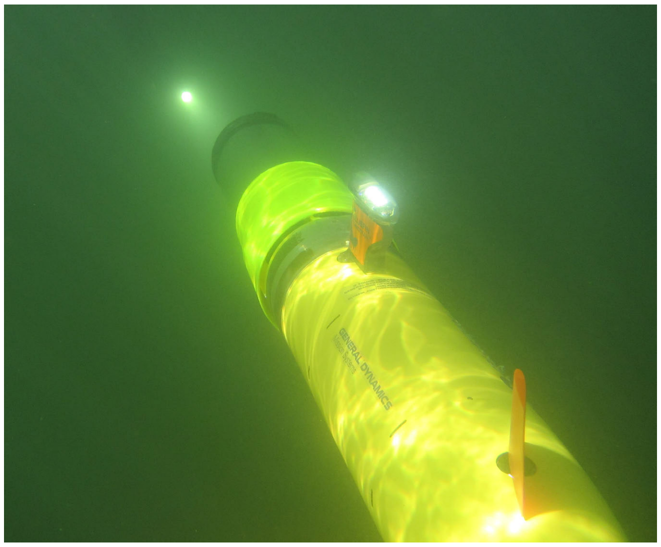


Fig. 5: SandShark with custom payload tracks a visual beacon while receiving commands from the backseat controller.

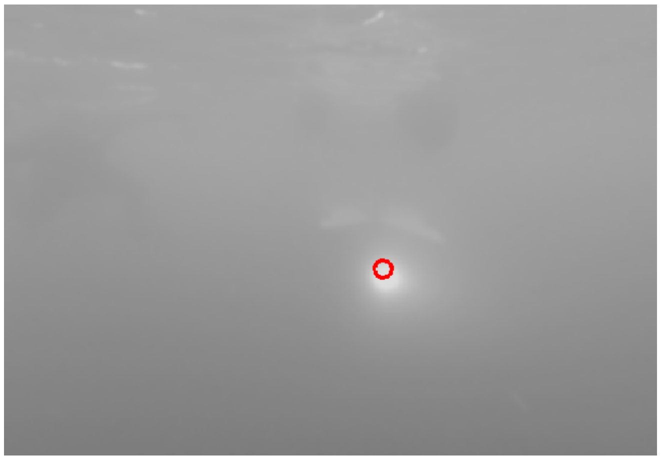
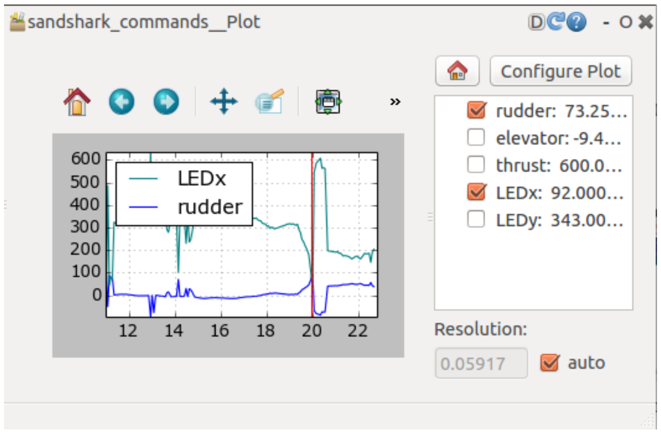


Fig. 6: OpenCV identifies the beacon. The result is bagged in ROS for review after the mission.

IV. Conclusion and Future Work

We have documented a frontseat-backseat control paradigm for Bluefin SandShark AUV. The frontseat controller is integrated in the commercial-off-the-shelf vehicle. Open-source hardware and software provide the framework for developing a versatile and efficient backseat controller. This backseat controller supports development of novel control algorithms in a dynamic testing environment. It has been validated in the pool environment, and is now being used during open-water testing.

This framework is ideal for application to many AUV platforms, including existing user-configurable vehicles [11], and for new vehicle concepts which implement adaptable behaviors. The authors plan to extend this work to the ROUGHIE vehicle [12], [13]. Additionally, the current state of the art in marine persistence compels a multi-vehicle, multi-modal, approach [14], which this framework enables. Although implementation of OpenCV methods has been smooth thus far, in the future a segmentation or classification method may be used that might over-tax the Raspberry Pi. In this event, a higher spec single board computer may be substituted. Finally, this development will facilitate extension of machine learning methods to AUV control and navigation tasks.


sandshark_commands__Plot

Configure Plot

- [✓] rudder: 73.25...
- [ ] elevator: -9.4...
- [ ] thrust: 600.0...
- [✓] LEDx: 92.000...
- [ ] LEDy: 343.00...

Resolution: 0.05917 [✓] auto

Fig. 7: Rqt_bag helps visualize command signals on a custom topic written to communicate with SandShark frontseat controller.

References

[1] N. R. Rypkema, E. M. Fischell, and H. Schmidt, “Closed-Loop Single-Beacon Passive Acoustic Navigation for Low-Cost Autonomous Underwater Vehicles,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[2] A. Melim and M. West, “Towards autonomous navigation with the Yellowfin AUV,” in *OCEANS’11 MTS/IEEE KONA*, Sep. 2011.

[3] D. P. Eickstedt and S. R. Sideleau, “The backseat control architecture for autonomous robotic vehicles: A case study with the Iver2 AUV,” in *OCEANS 2009*, Oct. 2009.

[4] T. Y. Teck and M. Chitre, “Hierarchical multi-agent command and control system for autonomous underwater vehicles,” in

[11] J. E. Manley and J. Smith, “Rapid development and evolution of a micro-UUV,” in *OCEANS 2017 - Anchorage*, Sep. 2017.

2012 IEEE/OES Autonomous Underwater Vehicles (AUV), Sep. 2012.

[5] O. A. Viquez, E. M. Fischell, N. R. Rypkema, and H. Schmidt, “Design of a general autonomy payload for low-cost AUV R D,” in *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, Nov. 2016.

[6] V. Djapic and D. Nad, “Using collaborative Autonomous Vehicles in Mine Countermeasures,” in *OCEANS 2010 IEEE - Sydney*, May 2010.

[7] D. Goldberg, “Huxley: A flexible robot control architecture for autonomous underwater vehicles,” in *OCEANS 2011 IEEE - Spain*, Jun. 2011.

[8] K. DeMarco, M. E. West, and T. R. Collins, “An implementation of ROS on the Yellowfin autonomous underwater vehicle (AUV),” in *OCEANS’11 MTS/IEEE KONA*, Sep. 2011.

[9] K. Sukvichai, K. Wongsuwan, N. Kaewnark, and P. Wisanuvej, “Implementation of visual odometry estimation for underwater robot on ROS by using RaspberryPi 2,” in *2016 International Conference on Electronics, Information, and Communications (ICEIC)*, Jan. 2016.

[10] B. Sütő, R. Dóczi, J. Kalló, B. Takács, T. A. Várkonyi, T. Haidegger, and M. Kozlovszky, “HSV color space based buoy detection module for autonomous underwater vehicles,” in *2015 16th IEEE International Symposium on Computational Intelligence and Informatics (CINTI)*, Nov. 2015.

[12] B. R. Page, S. Ziaeefard, A. J. Pinar, and N. Mahmoudian, “Highly Maneuverable Low-Cost Underwater Glider: Design and Development,” *IEEE Robotics and Automation Letters*, vol. 2, no. 1, Jan. 2017.

[13] S. Ziaeefard, B. R. Page, A. J. Pinar, and N. Mahmoudian, “Effective Turning Motion Control of Internally Actuated Autonomous Underwater Vehicles,” *Journal of Intelligent & Robotic Systems*, vol. 89, no. 1-2, Jan. 2018.

[14] T. Schneider and H. Schmidt, “Unified command and control for heterogeneous marine sensing networks,” *Journal of Field Robotics*, vol. 27, no. 6, May 2010.