Working toward the development of a generic marine vehicle framework: ROS-MVP

Emir Cem Gezer
College of Engineering
University of Rhode Island
Narragansett, RI, USA
emircem@uri.edu

Mingxi Zhou
Graduate School of Oceanography
University of Rhode Island
Narragansett, RI, USA
mzhou@uri.edu

Lin Zhao
College of Engineering
University of Rhode Island
Narragansett, RI, USA
linzhao@uri.edu

William McConnell
College of Engineering
University of Rhode Island
Kingston, RI, USA
wmcconnell@uri.edu

OCEANS 2022, Hampton Roads | 978-1-6654-6809-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/OCEANS47191.2022.9977346

Abstract—**Generic open-source software frameworks are significantly valuable for robotics research and development. With the intention of providing off the shelf AUV/ASV control, guidance, and operation solution, we developed a new open source framework called Robot Operating System Marine Vehicle Packages (ROS-MVP). The framework provides three sub-modules: a low-level controller, plugin-based behavior interface, and a mission planner. MVP mission planner comes with several common behaviors such as trajectory following and depth tracking. MVP is tightly integrated with ROS infrastructure and can be easily configured for different marine robots. In this paper, we present the details of ROS-MVP framework design and the field test results on an AUV in Narragansett Bay, Rhode Island.**

Index Terms—**Autonomous Underwater Vehicle (AUV), Autonomous Surface Vehicle (ASV), Open-source robotics, Marine robotics, Marine vehicle guidance and control**

I. Introduction

The emergence of unmanned marine vehicles is speeding up because of the increased availability of consumer-grade actuators, pressure housings, single board computers, among other components. Such advancements on the hardware side have enabled research groups from all around the world to come up with new Autonomous Underwater Vehicles (AUV) designs for various applications [1] [2] [3]. In the meantime, new software frameworks also have emerged in order to power the marine vehicles, decreasing the software development time drastically. Open-source simulation environments [4] [5] draw attention and create the possibility to kick-start marine robotics projects without needing a hardware platform. While these advancements have continued, many well-maintained small software projects grew bigger and stronger and became the building blocks for marine autonomy.

It is important for a framework to have a broad user base which ensures the healthy life-cycle of the software, for instances users can identify issues and provide contributions. Such frameworks also provide a unified programming interface, providing easy integration between different packages within its ecosystem. For instance, the Robot Operating System (ROS) has been powering a wide variety of robots [6] since its release. The ROS middleware has provided basic interfaces for easy software adoption and integration, accelerating the robotics development. The ros_control [7] and robot_localization packages are a good examples.

This work is supported by the National Science Foundation (NSF) under the award #2154901, RI-STAC Grant, and the Graduate School of Oceanography, University of Rhode Island.

Marine vehicles usually use a similar types of actuators, excluding the soft-bodied vehicles. These include propellers, thrusters, control surfaces, etc. The similarity within actuators can be exploited to create a control algorithm that fits most scenarios with different configurations. For example, control allocation methods use a specific approach, therefore, marine vehicles with various actuators can be controlled in the same approach using different control allocation matrices. Furthermore, vehicle navigation and guidance problems are well-developed with standard approaches. For example, robot localization package [8] is widely used in the robotics community, and the line-of-sight guidance law [9] is commonly used in marine robots for trajectory tracking [10].

To our best knowledge, there is no generic ROS compatible marine vehicle Guidance Navigation and Control (GNC) framework that is open-source and freely accessible. Therefore, we started the development of the ROS-MVP, an open-source (General Public License version 3 license) marine vehicle framework to fill the current gap. The rest of the paper is organized as follows. In the section II related work in marine robot software framework is reviewed. In section III, we describe the ROS-MVP design in details. We present our experiment results in section IV. Finally, we conclude the paper and discuss the future work in section V.

II. Related Work

There are several existing marine vehicle guidance, navigation, and control software middlewares running on different kinds of marine robots. The MOOS-IvP [11] is the most notable one. It comprises two main components: the MOOS [12] for inter-process communication (IPC), and IvP Helm for guidance and autonomy. It also has sensor fusion capabilities for localization through `pNav`, a navigation stack program. MOOS-IvP allows users to write custom software while keeping the existing packages for their ecosystem by using the Mission Oriented Operating Suite (MOOS). The work [13] is a good example of using the MOOS-IvP, where the authors successfully customized the MOOS-IvP framework and implemented a vessel tracking application with a Bluefin-21 AUV. Moreover, in [14], an autonomy payload for Bluefin Sandshark AUV was developed using MOOS-IvP framework. However, it does not have a generic vehicle controller, users of MOOS-IvP must write their own control software for their vehicle. In addition to that, it is not compatible with ROS due to their fundamental differences. Therefore, advancements in the ROS ecosystem can not be reflected easily in the MOOS-IvP ecosystem, and the gap is widening between them as robotics communities lean towards ROS [6]. Although there are some efforts to bridge these two frameworks together, such as the MOOS-IvP ROS bridge [15], combining two different architectures, remains to be a challenging task.

Furthermore, COLA2 [16] is a notable ROS-based framework specialized on AUVs [17]. COLA2 has the full-stack guidance, navigation and control solutions and currently running on the two AUVs, Sparus II and Girona 500, for a various missions [18]. Nonetheless, despite the fact of being an open-access project, COLA2 is a proprietary software that hampers the possibility of it becoming a defacto standard for ROS-based AUV control software.

Besides the MOOS-IvP and COLA2, some robotics simulation projects come with vehicle control, guidance and localization packages. For instance, UUV Simulator [4] project contains a simple thruster control allocation implementation, and its successor, Project DAVE [19], has more features, such as localization stack and perception sensor simulators available. Finally, using the vehicle controller inside the UUV Simulator imposes a high coupling problem that increases the complexity of the maintenance process, not to mention the fact that it is an unmaintained project as of today.

III. Design

A. Overview to the Framework

Motivated by the limitations on existing marine robotic frameworks, we started the development of a new framework called ROS-MVP, essentially, a GNC system. As shown in the figure 1, the proposed framework has three main components: a low-level vehicle controller (called MVP-Controller), a state-oriented behavior-based mission planner (called MVP-Helm), and the navigation system developed using the robot localization package [8]. The MVP-Controller is responsible for computing a set of actuator commands to satisfy the requested vehicle pose (position, velocity or orientation). It runs a control allocation method [20] with quadratic programming optimization under the hood. Then, The MVP-Helm process is responsible for guidance by utilizing behaviors. It dispatches desired vehicle pose to the MVP-Controller using guidance laws, such as the line-of-sight path following. The MVP-Helm and the MVP-Controller communicate over ROS topics. It is possible to replace the MVP-controller with a custom solution, e.g., a model-predictive controller or fuzzy logic controller, with the same ROS topics and services setup.

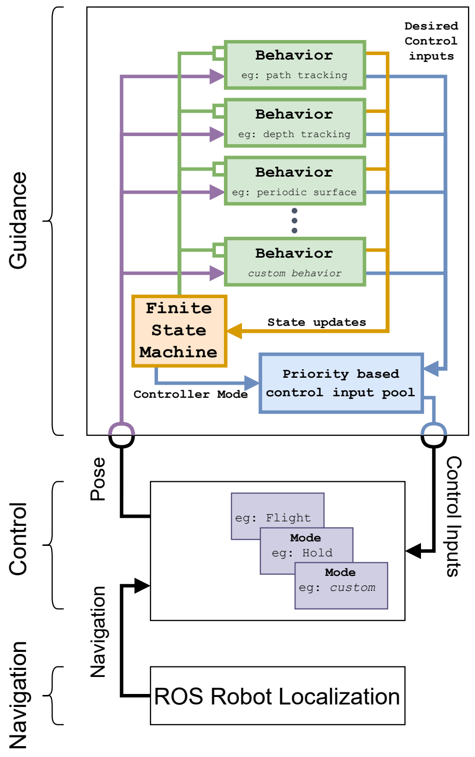

Fig. 1. MVP Architecture overview

B. Low-Level Controller design

The MVP-Controller uses a control allocation method to compute the desired forces for the actuators. The flow diagram for the MVP-Controller is shown in Fig.2. It works by first computing the necessary force and torque that should act on the body frame to achieve the desired pose using a Multiple Input Multiple Output Proportional Integral Derivative (MIMO-PID) controller for each degree of freedom (DOF). Then it feeds the target body frame force and torque to the control allocation matrix and ignores the DOFs that are not controlled in the current controller mode (configured by the user). Then, it applies the quadratic programming solver [21] to find the optimum set of forces for the thrusters to match the requested force and torque. Finally, the actual thruster command is solved based on the given thruster curves defined as polynomial functions.

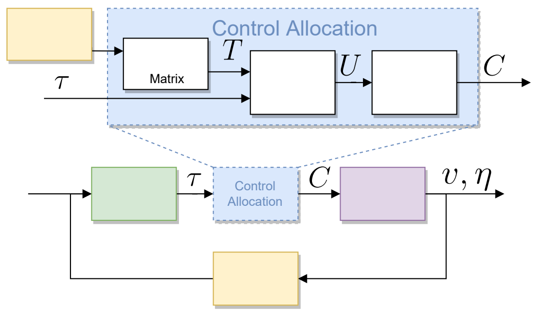

Fig. 2. MVP Controller flow diagram

MVP-Controller is configured via a YAML file where user could define the PID gains for each control mode and DOF, thrust curves for each thruster, relevant transform tree link names, and the odometry topic source. To make the PID tuning convenient, gains can be configured dynamically on the fly using the ROS dynamic reconfigure mechanism and its RQT plugin. Thrust curves that describe the force generated by the propulsion systems for a given control input are configured as general polynomials.

1) Control Law: The MVP-Controller uses the MIMO-PID controller. It computes the resultant force and torque, $\boldsymbol{\tau}$, needed on the vehicle for each DOF. The controller uses the feedback from the vehicle pose, $\boldsymbol{\eta} = [x, y, z, \phi, \theta, \psi]$ and $\boldsymbol{v} = [u, v, w, p, q, r]^\top$ using the SNAME 1950 notations [22].

2) Control Allocation: As indicated in Eq. 1, controller allocation matrix, $\boldsymbol{T}$ is used to project the forces from individual actuators, $\boldsymbol{U}$, into the forces and torques, $\boldsymbol{\tau}$, in the body frame. The elements in each column in $\boldsymbol{T}$ indicates the contribution of the forces and torques from each actuator. Then the contribution vectors were concatenated to create the control allocation matrix $\boldsymbol{T} = [t_1, t_2 ..., t_n]$.

$$\boldsymbol{\tau} = \boldsymbol{TU} \tag{1}$$

To find the optimal values in $\boldsymbol{U}$ such that the resulted forces and torques, $\boldsymbol{\tau}$, matches the required values, $\boldsymbol{\tau}^*$, from the control law as close as possible, we define an objective function as shown in Eq.2 which is the sum of the squares of the difference between $\boldsymbol{\tau}^*$ and $\boldsymbol{\tau}$

$$\begin{aligned} J =& (\boldsymbol{TU} - \boldsymbol{\tau}^*)^\top (\boldsymbol{TU} - \boldsymbol{\tau}^*) \\ =& \boldsymbol{U}^\top \boldsymbol{T}^\top \boldsymbol{TU} - \boldsymbol{U}^\top \boldsymbol{T}^\top \boldsymbol{\tau}^* - \boldsymbol{\tau}^{*\top} \boldsymbol{TU} + \boldsymbol{\tau}^{*\top} \boldsymbol{\tau}^* \\ =& \boldsymbol{U}^\top \boldsymbol{T}^\top \boldsymbol{TU} - 2\boldsymbol{\tau}^{*\top} \boldsymbol{TU} + \boldsymbol{\tau}^{*\top} \boldsymbol{\tau}^* \end{aligned} \tag{2}$$

As shown in Eq.2, the objective function can be further expanded into a similar format as the standard quadratic programming (QP) problem expressed in Eq.3 and 4 where the symbol replacements are indicated and the inequality constraints are used to limit the resulting thrusts for each thrusters. Such a QP problem is solved using the OSQP solver [21] in the MVP-Controller.

$$\begin{aligned} &\min_{x} \ J = (\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x}) \\ &\text{subject to } \boldsymbol{Ax} \leq \boldsymbol{b} \end{aligned} \tag{3}$$

$$\begin{aligned} &\min \underbrace{\boldsymbol{U}^\top}_{\boldsymbol{x}^\top} \underbrace{\boldsymbol{T}^\top \boldsymbol{T}}_{\frac{1}{2}\boldsymbol{Q}} \underbrace{\boldsymbol{U}}_{\boldsymbol{x}} + \underbrace{(-2\boldsymbol{\tau}^{*\top} \boldsymbol{T})}_{\boldsymbol{c}^\top} \underbrace{\boldsymbol{U}}_{\boldsymbol{x}} \\ &\text{subject to } \boldsymbol{AU} \leq \boldsymbol{b} \end{aligned} \tag{4}$$

Thrust curves describe the characteristics of the thruster with respect to the control commands (e.g., PWM signals). In MVP-Controller, thrust curves are defined using general polynomials. A polynomial solver from GNU Scientific Library [23] computes the control command $\boldsymbol{C} = [c_1, c_2, ..., c_n]^\top$, after QP solver has obtained the optimum forces, U, for each thrusters.

3) Configuration: The control allocation matrix can be configured in two ways, either manually entered or automatically computed from the given transform tree link names of the actuators. The transform tree based thruster allocation matrix generation works by computing the distances and rotation to an defined center of gravity link. To acquire body frame force and torque contribution per actuator for the allocation matrix $\boldsymbol{T}$, the distance and the rotation of an actuator to a center of gravity link were computed.

In the MVP-Controller, user can define different *control modes*, and in each mode user can decide which DOF to control. For example, in the *flight* mode, the surge velocity and yaw and pitch angles are controlled, and in the *hold* mode, the X, Y, and Z position of the vehicle are controlled. For the same DOF, user could also define different PID gains in different modes. For instance, the PID gains for heading can be different in surge and hovering modes as the vehicle's regions of operation are different.

C. Mission planner design

The MVP mission planner is designed to manage behaviors using a finite state machine (FSM). The main actor in the MVP mission planner is a ROS node called MVP-Helm. The mission planner exploits the MVP controller's ability to separately control each DOFs. Each behavior is responsible for generating control inputs for DOFs that they are programmed to control. In addition to that, behaviors can trigger an FSM state change. MVP Mission stack provides an abstract C++ class via Pluginlib package in ROS middleware for behavior development, so the users of MVP mission planner can introduce their behavior by using that library.

Figure 3 explains how the MVP-Helm node handles the control inputs from active behaviors. As shown in Fig.3, MVP-Helm is configured to have two states; "survey" and "start". The "survey" state controls the vehicle in flight mode ($M_{flight} = [u, \psi, \theta]$) while the "start" uses the controller in hold position mode ($M_{hold} = [x, y, z]$). As indicated by different colors in Fig. 3, the vehicle is currently in the "survey" state where path tracking, depth tracking, and the periodic surfacing behaviors are active. By design, both depth tracking and periodic surfacing behaviors controls the pitch $\boldsymbol{theta}$. To avoid conflicts, we have designed a priority-based DOF selector to select the desired pose from different behaviors based on the priorities. For example, when the periodic surfacing behavior is activated using a timer, the MVP-Helm will send the desired pitch angle from the periodic surfacing behavior rather than that from the depth tracking behavior to the MVP-Controller because the periodical surfacing has a higher priority rank ($p = 2$). We, currently, have implemented several commonly used behaviors such as line-of-sight path tracking, depth tracking and periodic surfacing, etc.

IV. Results

The MVP is deployed on the recently developed prototype ALPHA AUV [24] shown in figure 4, and field trials were

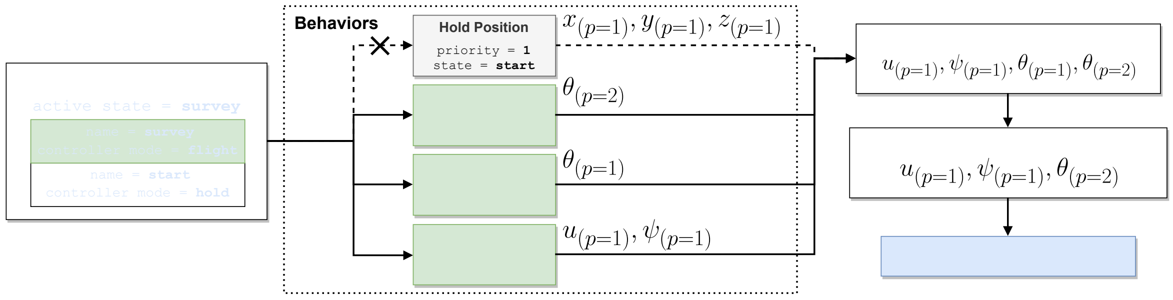

Fig. 3. MVP Helm flow diagram. Priorities for desired vehicle pose requested from behaviors are shown in letter p. Green boxes indicate active state and executed behaviors.

conducted to validate the performance. The ALPHA AUV has three BlueRobotics T200 thrusters; horizontal and vertical thrusters near the bow and one main thruster at the stern. It is equipped with a Waterlinked A50 DVL, Xsens MTI-630 AHRS (Attitude and heading reference system), and an Adafruit Ultimate GPS for navigation. It can be operated with and without a tether cable. Prior the field tests in the pond and in Narragansett Bay, Rhode Island., several missions and control scenarios were executed in the simulation to test the premise of the MVP architecture.

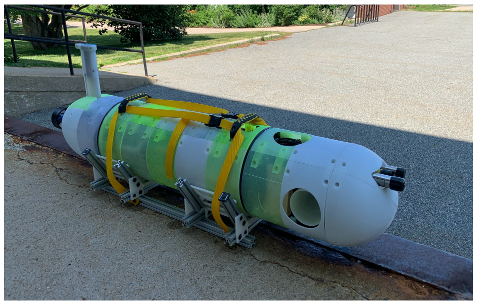

Fig. 4. ALPHA AUV at the loading dock of the Graduate School of Oceanography University of Rhode Island. The two tunnel thrustes can be seen on the nose and the main thruster on the stern side. The mast on the stern end of the vehicle is equipped WiFi, long range radio, and GPS.

A. Simulation Results

The simulation environment were developed leveraging the Stonefish simulator [5]. In the figure 5, test results of MVP architecture from the Stonefish simulator are shown. The conducted mission is a similar mission that is shown in the figure 3. The vehicle starts its mission by diving to 2 meters depth and moving towards the first goal point $(x = 0, y = 0)$. When the vehicle approaches the 3^{rd} goal point $(x = 20, y = 20)$, the periodic surfacing is triggered and the behavior commands the vehicle to climb up to the surface. After surfacing was done within 10 seconds, the vehicle dives to the desired depth and continues its survey. Note that during the simulation tests, the simulated vehicle was equipped with two vertical thrusters; one in the stern and one in the bow.

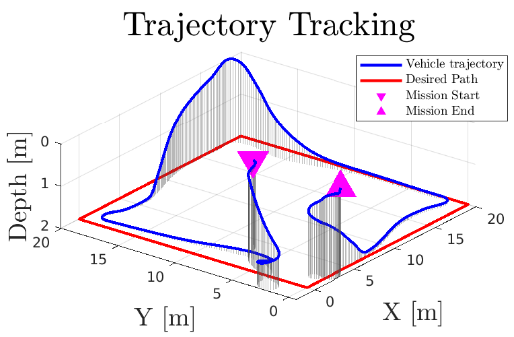

Fig. 5. Path tracking performance from the Stonefish simulator

B. Field test Results

Several field tests were conducted at Beach Pond, RI, USA, and Narragansett Bay, RI, USA, to validate the MVP architecture. In both of the tests, the vehicle was programmed to conduct path-tracking and depth-following behaviors. The programmed path was a 50m-by-50m square and the depth was 3 meters, and the result of one run is shown in 6. The vehicle was able to follow the programmed path. However, the depth tracking performance was poor due to improper PID gains and ocean currents. In this mission periodic surfacing behavior was disabled.

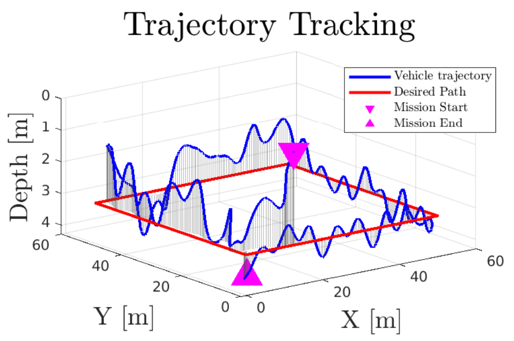

Fig. 6. Path tracking performance from the field test conducted at Narragansett Bay.

V. Conclusion and Future work

In this paper, a generic ROS-based software framework, ROS-MVP, is presented for marine robotic platforms, such as AUVs and ASVs. The introduced architecture has three main components; a low-level controller (MVP-Controller), a mission planner (MVP-Helm), and behaviors. The first component, MVP-Controller, provides a generic multiple DOF vehicle controller using MIMO-PID with QP optimization. It requires odometry information from a navigation source, such as the ROS robot localization package. The second component, MVP-Helm, pilots the vehicle by executing behaviors and sending their outputs to the vehicle controller. A finite state machine decides behavior execution and a priority pool selects the actions with the highest priority from the behavior results. The last component, MVP behaviors, are customizable and are managed and executed by the MVP-Helm. In order to validate the system performance and functionality, simulations and field tests have been conducted using ALPHA AUV, and the results were presented.

The proposed framework is planned to be further tested on an ASV platform. The research team successfully integrated MVP framework to Heron ASV (from Clearpath Robotics) in the simulation environment, and will move forward with hardware integration.

The current version of the MVP architecture has several limitations. First, it currently doesn't support control surfaces such as fins and masts, and azimuth thrusters. The availability of testing platforms is the main reason behind this limitation. However, within the development process, the Stonefish [5] simulator is found to be a sufficient sandbox environment for developing and testing such a feature. Secondly, MVP is only tested in Ubuntu 20.04 operating system and is only compatible with ROS1. As the robotics community migrates to ROS2 [25], it remains to be a crucial limitation. The research team has planned actions to resolve these two limitations in the near future. Overall, with these changes and further developments, we expect to provide an customizable ROS compatible GNC framework for marine vehicles. We hope to provide this vital resource to the community, and have it be openly accessible for new marine vehicle projects in the future.

References

[1] Chelsey Edge, Sadman Sakib Enan, Michael Fulton, Jungseok Hong, Jiawei Mo, Kimberly Barthelemy, Hunter Bashaw, Berik Kallevig, Corey Knutson, Kevin Orpen, et al. Design and experiments with loco auv: A low cost open-source autonomous underwater vehicle. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1761–1768. IEEE, 2020.

[2] Daniel A Duecker, Nathalie Bauschmann, Tim Hansen, Edwin Kreuzer, and Robert Seifried. Hippocampusx–a hydrobatic open-source micro auv for confined environments. In *2020 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*, pages 1–6. IEEE, 2020.

[3] Arron Griffiths, Aleksandr Dikarev, Pete R. Green, Barry Lennox, Xavier Poteau, and Simon Watson. Avexis—aqua vehicle explorer for in-situ sensing. *IEEE Robotics and Automation Letters*, 1(1):282–287, 2016.

[4] Musa Morena Marcusso Manhães, Sebastian A Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. Uuv simulator: A gazebo-based package for underwater intervention and multi-robot simulation. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–8. IEEE, 2016.

[5] Patryk Cieślak. Stonefish: An advanced open-source simulation tool designed for marine robotics, with a ros interface. In *OCEANS 2019 - Marseille*, pages 1–6, 2019.

[6] Lin Zhang, Robert Merrifield, Anton Deguet, and Guang-Zhong Yang. Powering the world's robots—10 years of ros. *Science Robotics*, 2(11):eaar1868, 2017.

[7] Sachin Chitta, Eitan Marder-Eppstein, Wim Meeussen, Vijay Pradeep, Adolfo Rodríguez Tsouroukdissian, Jonathan Bohren, David Coleman, Bence Magyar, Gennaro Raiola, Mathias Lüdtke, et al. ros_control: A generic and simple control framework for ros. *The Journal of Open Source Software*, 2(20):456–456, 2017.

[8] T. Moore and D. Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[9] Anastasios M Lekkas and Thor I Fossen. Line-of-sight guidance for path following of marine vehicles. *Advanced in marine robotics*, pages 63–92, 2013.

[10] Zhi Li, Ralf Bachmayer, and Andrew Vardy. Path-following control for unmanned surface vehicles. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4209–4216, 2017.

[11] Michael R Benjamin, Henrik Schmidt, Paul M Newman, and John J Leonard. Nested autonomy for unmanned marine vehicles with moos-ivp. *Journal of Field Robotics*, 27(6):834–875, 2010.

[12] Paul Michael Newman. Moos-mission orientated operating suite. 2008.

[13] Artur Wolek, James McMahon, Benjamin R. Dzikowicz, and Brian H. Houston. Tracking multiple surface vessels with an autonomous underwater vehicle: Field results. *IEEE Journal of Oceanic Engineering*, 47(1):32–45, 2022.

[14] Oscar A. Viquez, Erin M. Fischell, Nicholas R. Rypkema, and Henrik Schmidt. Design of a general autonomy payload for low-cost auv r&d. In *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 151–155, 2016.

[15] Kevin DeMarco, Michael E West, and Thomas R Collins. An implementation of ros on the yellowfin autonomous underwater vehicle (auv). In *OCEANS 2011*, pages 1–7. IEEE, 2011.

[16] Narcis Palomeras, Andres El-Fakdi, Marc Carreras, and Pere Ridao. Cola2: A control architecture for auvs. *IEEE Journal of Oceanic Engineering*, 37(4):695–716, 2012.

[17] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[18] Nuno Gracias, Pere Ridao, Rafael Garcia, Javier Escartín, Michel L'Hour, Franca Cibecchini, Ricard Campos, Marc Carreras, David Ribas, Narcís Palomeras, Lluis Magi, Albert Palomer, Tudor Nicosevici, Ricard Prados, Ramon Hegedüs, Laszlo Neumann, Francesco de Filippo, and Angelos Mallios. Mapping the moon: Using a lightweight auv to survey the site of the 17th century ship 'la lune'. In *2013 MTS/IEEE OCEANS - Bergen*, pages 1–8, 2013.

[19] Mabel M. Zhang, Woen-Sug Choi, Jessica Herman, Duane Davis, Carson Vogt, Michael McCarrin, Yadunund Vijay, Dharini Dutia, William Lew, Steven Peters, and Brian Bingham. Dave aquatic virtual environment: Toward a general underwater robotics simulator. In *2022 IEEE/OES Autonomous Underwater Vehicle (AUV) Symposium*, pages 1–8, 2022.

[20] Thor I. Fossen and Tor A. Johansen. A survey of control allocation methods for ships and underwater vehicles. In *2006 14th Mediterranean Conference on Control and Automation*, pages 1–6, 2006.

[21] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[22] Thor I Fossen. Guidance and control of ocean vehicles. *University of Trondheim, Norway, Printed by John Wiley & Sons, Chichester, England, ISBN: 0 471 94113 1, Doctors Thesis*, 1999.

[23] Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Patrick Alken, Michael Booth, Fabrice Rossi, and Rhys Ulerich. *GNU scientific library*. Network Theory Limited, 2002.

[24] Mingxi Zhou, Emir Cem Gezer, and William McConnell. Acrobatic low-cost portable hybrid auv (alpha): System design and preliminary results. In *OCEANS 2022: Hampton Roads*. IEEE, 2022.

[25] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.