Performance characterization and comparison of conventional and machine-learning-based techniques for control of a USV

Armando J. Sinisterra, Siddhartha Verma, and Manhar R. Dhanak
Department of Ocean and Mechanical Engineering
The Institute for Ocean & Systems Engineering
Florida Atlantic University
Boca Raton, USA
asiniste@fau.edu; vermas@fau.edu; dhanak@fau.edu

***Abstract*— This study is part of ongoing work on situational awareness and autonomy of a 16' WAM-V USV. The objective of the study is to assess the merits of application of two different path-following methods for the motion control of the USV, including characterizing and evaluating their performance, robustness and potential in actual field implementation. The first method uses a conventional control technique based on linear theory, while the second method uses Reinforcement Learning. The performances of the two methods for selected case studies are compared and discussed.**

Keywords—controls, machine-learning, marine robotics, artificial intelligence, unmanned surface vehicles.

I. INTRODUCTION

Unmanned Surface Vehicle (USVs) applications are increasingly expanding, and range from search and rescue to research-related and exploration operations. The level of autonomy of these marine vehicles depends on the interaction between its ability to model the world through data acquired with perception sensors (such as RADAR, LiDAR, etc.), and to control the dynamics involved (external forces acting on the vehicle), based on a series of goals (waypoints or tasks) computed by a high-level mission planner (which includes path-planning), preventing the vehicle to interact with the environment in undesirable ways (collisions), while leading the vehicle to a desired pose. This work is focused on the latter, specifically, on the motion control of a 16 ft WAM-V (Wave Adaptive Modular Vehicle) USV over consecutive waypoints conforming a path.

There are mainly two primitive control modes in USV operations. The first one consists of controlling a fixed position and orientation of the vehicle, namely station-keeping control [1]. The second mode, which is the focus of this work, consists of controlling the motion of the vehicle along a given path, while minimizing errors corresponding to deviations from a desired state. This desired state depends on what the control variables are and what type of controller is to be used. When using a trajectory tracking controller, for example, the desired state can be given in terms of the pose of the vehicle and its first derivative with respect to time. This approach, along with prior trajectory planning data [2], controls the vehicle to follow this path, while minimizing the tracking error associated to the state vector [3]. One of the advantages of this method is that it takes into account the dynamic limitations of the vehicle itself (maximum velocity, acceleration, and turning rate), thus ensuring smooth transitional maneuvers, like when changing the bearing due to corner-like turns in a succession of waypoints.

A number of conventional control methods have been implemented in order to control the motion of USVs. Oftentimes, a control approach based on linear control theory is used as a benchmark against more involved methods based on nonlinear control theory, such as, e.g., feedback linearization and Backstepping [4] as well as Sliding Mode control [5] techniques, which constitutes in general what is known as trajectory tracker controllers.

Resultant control laws from conventional control methods, usually impose an additional challenge, which translates on how to transform the control forces and moments computed by the controller (typically applied at the center of gravity of the USV) into actual thrust and/or rotation of the thrusters involved in the USV configuration. This stage is known as control allocation, and most suitable methods depend on the thruster configuration of the vehicle, such as whether the vehicle is underactuated or fully-actuated, as well as on the degrees of freedom of the thrusters involved as with fixed or azimuth thrusters [6].

Lately, machine-learning (ML)-based algorithms for motion control of autonomous vehicles is gaining interest. Here the agent (the vehicle) is trained either in real-life [7] or in a simulated environment [8], (or both), over many iterations, so that the algorithm learns a control policy which generates the corresponding actions to conduct the vehicle to its desired state. Unlike conventional control methods, this approach does not need to address the control allocation problem, given that the training is based directly on analysis of the thruster responses in generating the desired motions.

This work explores and characterize both a conventional control technique and a machine-learning based method for path-following control of a particular USV that consists of two transom azimuth (electric) thrusters. The first technique consists of a multi-input multi-output (MIMO) PD path-following controller, similar to [9], that uses three independent PD controllers for simultaneously controlling the surge speed, cross-track error (relative to the desired path), and heading of the USV. It also implements a quadratic program (QP) optimization approach in order to address the control allocation problem. The second is a machine-learning control method, based on the reinforcement learning (RL) technique, implementing Artificial Neural Networks (ANN) for determining the optimal policy. The goal is to train the agent (USV) using the vehicle's 3-

This work is supported by the Office of Naval Research under grants N000141512724 and N00014-18-1-2212 (Program Manager: Kelly Cooper).

978-0-578-57618-3 ©2019 MTS

degree-of-freedom (3DOF) motion model so that the algorithm learns a control policy which generates the corresponding actions (speed and steering of the thrusters), in order to conduct the USV to travel along a straight path between two consecutive waypoints (notice that in general a path can be considered as a sequence of straight lines). Both methodologies make use of the 3DOF motion model as described in [1], with some modifications, in order to govern the motion of the USV in a simulated environment.

II. WAM-V 16' USV

A. Description

The 16 ft. Wave Adaptive Modular Vehicle (WAM-V) platform (Fig. 1) manufactured by Marine Advanced Robotics [10], offers a convenient catamaran-type surface vehicle for USV applications. It provides an overall lightweight structure with a high payload to weight ratio, allowing for accommodation of all the systems required onboard the vehicle while facilitating handling in field operations.

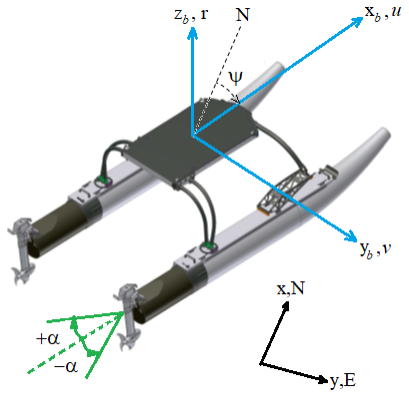

Fig. 1. WAM-V inertial and body-fixed coordinates frames defined in north-east-down (NED) convention as (x, y) and (x_b, y_b) respectively. Surge, sway and yaw velocities (u, v, r) are correspondingly aligned with the body-fixed coordinates, and thrusters azimuth angles are defined as α

Its center tray is connected on each side to the corresponding pontoon through a set of articulated bars and a suspension system. This arrangement minimizes the motion and vibration effects of incident waves over the center tray, making it ideal for the data acquisition tasks of the sensors onboard the WAM-V.

The vehicle is equipped with two electric motors with an input power of 2 kW, each coupled with a linear actuator which jointly conform a pair of transom azimuth thrusters, restricted to an azimuth rotation between -45° and 45° (α in Fig. 1).

B. Equations of Motion

The dynamic model used in both control methods for training, simulations and results presented in this work use the SNAME 1950 representation for the surge, sway and yaw degrees of freedom (3DOF). Eq. (1) corresponds to the kinematic transformation of velocities in 3DOF (as illustrated in Fig. 1) from body-fixed to inertial coordinates, where $\boldsymbol{J}(\psi)$ is the transformation (rotation) matrix. The 3DOF kinetic model on the other hand is represented by Eq. (2), where $\boldsymbol{M}$ is the inertial matrix (Eq. (3)); $\boldsymbol{C}(\boldsymbol{\nu})$ represents the Coriolis and centripetal terms (Eq. (4)). Both $\boldsymbol{M}$ and $\boldsymbol{C}(\boldsymbol{\nu})$ account for the rigid-body and added mass effects. $\boldsymbol{D}(\boldsymbol{\nu})$ is the drag matrix for which only the linear components are taken into account (Eq. (5)); $\boldsymbol{\tau}$ is the generalized force vector (environmental forces are not included in this model) produced jointly by the thrusters according to the particular configuration of the vehicle, where T_x and T_y are the thrust components aligned along the corresponding body-fixed coordinates and M_z is the resulting moment around the z_b axis.

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\psi)\boldsymbol{\nu} \tag{1}$$

Where,

$$\dot{\boldsymbol{\eta}} = [\dot{x}\ \dot{y}\ \dot{\psi}]^T$$

$$\boldsymbol{\nu} = [u\ v\ r]^T$$

$$\boldsymbol{J}(\psi) = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}\boldsymbol{\nu} = \boldsymbol{\tau} \tag{2}$$

$$\boldsymbol{M} = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & 0 \\ 0 & 0 & I_z - N_{\dot{r}} \end{bmatrix} \tag{3}$$

$$\boldsymbol{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -(m - Y_{\dot{v}})v \\ 0 & 0 & (m - X_{\dot{u}})u \\ (m - Y_{\dot{v}})v & -(m - X_{\dot{u}})u & 0 \end{bmatrix} \tag{4}$$

$$\boldsymbol{D} = -\begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & 0 \\ 0 & 0 & N_r \end{bmatrix} \tag{5}$$

$$\boldsymbol{\tau} = [T_x\ T_y\ M_z]^T \tag{6}$$

The origin of the body-fixed coordinate frame is located at the center of gravity of the vehicle and the mass properties and hydrodynamic parameters have been slightly updated from [1].

III. Path Following Control

The path-following controller works towards minimizing the state error at the sample rate of the system. The state vector is given by the surge speed (u), cross-track error (cte) and heading (ψ) of the vehicle. The control law is determined by a multi-input multi-output (MIMO) linear controller, where each state variable is being controlled independently by a PD controller. A methodology described below is followed in order to compute the cte and the desired heading(ψ_d), as well as to address the control allocation problem for the thruster configuration of this particular USV.

A. MIMO Path-Following PD-Control

The methodology is centered around conducting the vehicle, at a predefined cruise-speed, along a certain path (defined as a sequence of straight lines, each entailed between two consecutive points) by minimizing the cte variable directly, and also by means of following an instantaneous desired heading (ψ_d) using the enclosure-based steering method [6], [11]. The output of this MIMO controller is a vector of forces in the surge and sway directions and a moment in the yaw direction, similar to Eq. (6), and is referred to as τ_c in Fig. 2. This figure depicts the entire path-following control flow diagram used in computational simulations, where the deviations between the current state of the vehicle from its desired state is translated into

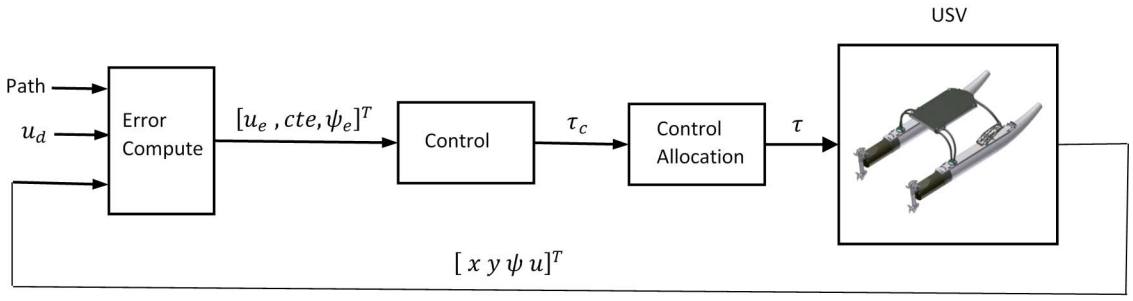

Fig. 2. Path-following control diagram for simulation

control forces which need further allocation in order to consider the actual actuators onboard the USV, accounting for their dynamic constraints. The control variables are define below:

- The surge speed is defined as the desired cruise-speed of the vehicle, for which a value between 1.5 and 5 *m/s* is reasonable in typical USV operations. This variable is controlled by the first of the three PID controllers which translates the surge speed error into a corresponding surge force.
- The cross-track error (*cte*) is defined as the smallest (orthogonal) distance between the reference point of the vehicle (in this case located at its center of gravity) and the desired path, approximated in the vicinity as a straight line between points $\vec{p}_k$ and $\vec{p}_{k+1}$ in Fig. 3. This distance is computed as the norm of vector $\vec{v}$ according to Eq. (9) and (10), defined in turn as a function of vectors $\vec{R}$ and $\vec{u}$ which correspond to the position of the vehicle with respect to $\vec{p}_k$, and its projection over the path segment (Eq. (7)), respectively. The magnitude of vector $\vec{u}$ (Eq. (8)) dictates when the algorithm needs to switch to the next path segment, that is, if $\|\vec{u}\| \geq \|\vec{S}\|$ consider instead the next path segment in the sequence in order to compute the *cte*. The variable *cte* is controlled by a second PID controller which translates the cross-track error value into a corresponding sway force.

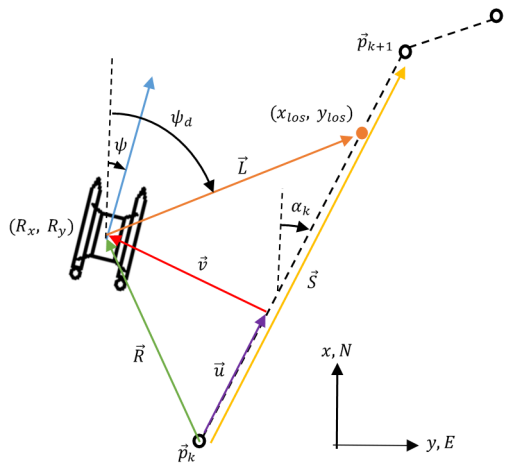

Fig. 3. Path-following geometry

$$\vec{u} = \|\vec{u}\|\hat{S} \tag{7}$$

$$\|\vec{u}\| = \vec{R}.\hat{S} = \frac{R_x S_x + R_y S_y}{\sqrt{S_x^2 + S_y^2}} \tag{8}$$

$$\vec{v} = \vec{R} - \vec{u} \tag{9}$$

$$\|\vec{v}\| = \frac{S_x R_y - S_y R_x}{\sqrt{S_x^2 + S_y^2}} = cte \tag{10}$$

- The desired heading (ψ_d) in Eq. (11) is defined according to the enclosure-based steering method in [6]. It computes a heading value which allows the USV to smoothly minimize the *cte* while travelling along the path, and it is a function of coordinates x_{los} and y_{los} which are computed using Eq. (12) and (13), where $\vec{p}_k = (x_k, y_k)^T$ and $\vec{p}_{k+1} = (x_{k+1}, y_{k+1})$. Vector $\vec{L}$, shown in Fig. 3, is a vector defining ψ_d with magnitude equal to R which corresponds to the radius of a circle centered at the vehicle (Eq. (12)), large enough to intersect the current path segment on two different points, and set as a predefined parameter representing a look-ahead distance over the desired path. Also, α_k is the angle of the path segment with respect to *x* (North). The heading variable ψ is then controlled by a third PID controller which translates the heading error ψ_e into a corresponding yaw moment.

$$\psi_d = atan2\left(\frac{y_{los} - y}{x_{los} - x}\right) \tag{11}$$

$$(x_{los} - x)^2 + (y_{los} - y)^2 = R^2 \tag{12}$$

$$\tan(\alpha_k) = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} = \frac{y_{los} - y_k}{x_{los} - x_k} = constant \tag{13}$$

B. Constrained Nonlinear Iterative Control Allocation Using Quadratic Programming

The nonconvex optimization problem entailed by the control allocation of azimuth thrusters in marine crafts can be reformulated as a locally convex quadratic programming (QP) optimization problem, to be solved iteratively within every time step; this approach is formulated in Eq. (14), similar to [12] after some mathematical manipulations.

Vector $[\Delta \boldsymbol{f}, \Delta \boldsymbol{\alpha}, \boldsymbol{s}]$ contains the variables to be optimized, so as to produce the minimum cost, where $\boldsymbol{f} = \boldsymbol{f}_o + \Delta \boldsymbol{f}$ corresponds to the current force vector (including all thrusters),

defined as the sum of the last force $\boldsymbol{f}_o$ and the increment $\Delta\boldsymbol{f}$ computed after optimization. Similarly, the current azimuth angles are defined by the vector $\boldsymbol{\alpha} = \boldsymbol{\alpha}_o + \Delta\boldsymbol{\alpha}$, where $\Delta\boldsymbol{\alpha}$ ensures the azimuth angles do not move too much from the previous sample $\boldsymbol{\alpha}_o$, according to constraint (18). Vector $\boldsymbol{s}$ represents the error between $\boldsymbol{\tau}_c$ and $\boldsymbol{\tau}$ (see Fig. 2), that is, between the goal and the achieved control forces.

$$C = \min_{\Delta\boldsymbol{f},\,\Delta\boldsymbol{\alpha},\,\boldsymbol{s}} \left\{ \Delta\boldsymbol{f}^T\boldsymbol{P}\Delta\boldsymbol{f} + \boldsymbol{s}^T\boldsymbol{Q}\boldsymbol{s} + \Delta\boldsymbol{\alpha}^T\boldsymbol{\Omega}\Delta\boldsymbol{\alpha} + \mathbf{f}_o^{\mathrm{T}}2\boldsymbol{P}\Delta\boldsymbol{f} + \frac{\partial}{\partial\boldsymbol{\alpha}}\left(\frac{\varrho}{\varepsilon + \det(\boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{T}^T(\boldsymbol{\alpha})}\right)_{\boldsymbol{\alpha}_o}\Delta\boldsymbol{\alpha} \right\} \tag{14}$$

Subject to:

$$\boldsymbol{s} + \boldsymbol{T}(\boldsymbol{\alpha}_o)\Delta\boldsymbol{f} + \frac{\partial}{\partial\boldsymbol{\alpha}}(\boldsymbol{T}(\boldsymbol{\alpha}_o)\boldsymbol{f})|_{\boldsymbol{\alpha}_o,\boldsymbol{f}_o}\Delta\boldsymbol{\alpha} = \boldsymbol{\tau} - \boldsymbol{T}(\boldsymbol{\alpha}_o)\boldsymbol{f}_o \tag{15}$$

$$\boldsymbol{f}_{min} - \boldsymbol{f}_o \le \Delta\boldsymbol{f} \le \boldsymbol{f}_{max} - \boldsymbol{f}_o \tag{16}$$

$$\boldsymbol{\alpha}_{min} - \boldsymbol{\alpha}_o \le \Delta\boldsymbol{\alpha} \le \boldsymbol{\alpha}_{max} - \boldsymbol{\alpha}_o \tag{17}$$

$$\Delta\boldsymbol{\alpha}_{min} \le \Delta\boldsymbol{\alpha} \le \Delta\boldsymbol{\alpha}_{max} \tag{18}$$

The first and fourth terms in (14) correspond to the quadratic and linear power consumption terms, respectively. A quadratic polynomial fit of the propulsive power expression in Eq. (19) was performed in order to get the corresponding diagonal power matrix $\boldsymbol{P} \in \mathbb{R}^2$ (since we have two thrusters) [13].

$$P(T) = (P_{max} - P_{min})\left(\frac{|T|}{T_{max}}\right)^{\eta} + P_{min} \tag{19}$$

Where,

$P_{max} = 1120\ W$, max. propulsive power per motor [14]

$P_{min} = 0\ W$;

$T_{max} = 350\ N;\ \ T_{min} = -250\ N$;

$1.3 \le \eta \le 1.7$, typically.

The second term in (14) penalizes the error $\boldsymbol{s}$ described previously, in order to guarantee that there is an optimal solution for any $\boldsymbol{\tau}$ and $\boldsymbol{\alpha}_o$. Positive definite matrix $\boldsymbol{Q}$ is chosen so that $\boldsymbol{s} \approx \mathbf{0}$, with weights so large that constraint (15) is satisfied whenever possible [6].

The third term, along with constraints (17) and (18), minimize the rate of change in azimuths while constraining them to change no more than it is determined by the mechanical capacity of the thruster, for which matrix $\boldsymbol{\Omega} > 0$ is used to tune this objective [12]. This term also accounts for the 'danger zone' in which the azimuth is close to the thruster's limit of rotational capacity, which for this particular case is within the range of $[-45°, 45°]$, so there might be cases where $\Delta\boldsymbol{\alpha}$ can be smaller than $|\Delta\boldsymbol{\alpha}_{max}|$ or $|\Delta\boldsymbol{\alpha}_{min}|$ as it is reflected in (17). The azimuth rate was determined according to experimental data, in which an entire turn from $-45°$ to $45°$ was measured to be approximately 2 seconds in the actual USV. Knowing that navigational sensor data is available at a frequency of 100 Hz (sample rate of 0.01 seconds), we calculate how much the thruster is limited to rotate within the sample rate, for a result of $0.45°$ per sample or $(0.0079\ rad/0.01\ sec)$. This value of $0.0079\ rad$ is then used to define $\Delta\boldsymbol{\alpha}_{min}$ and $\Delta\boldsymbol{\alpha}_{max}$ in Eq. (18).

Maximum and minimum thrust values acquired from bollard-pull tests [9] are accounted for in Eq. (16), while thrust-rate is handled by low-pass filtering the thrust values in order to get a more realistic response. For this, Newton's second law was used to describe the response of the electric motor as shown in Eq. (20), where J_R is the moment of inertia of the rotor, b is a damping constant and T is the torque generated by the motor. The transfer function (angular speed over torque) was then computed in the Laplace domain, as shown in (21), obtaining the same form as the continuous low-pass filter by making $b = 1$.

$$J_R\ddot{\theta} + b\dot{\theta} = T \tag{20}$$

$$\frac{s\Theta(s)}{T(s)} = \frac{1}{J_R s + b} \tag{21}$$

Provided that the thrust in a propulsion system based on an electric motor depends on the torque T generated, the value of J_R was also used as the time constant for low-pass filtering the thrust as an approximation to model the dynamics of the actuators in the simulation model. An actual value of J_R was used from a motor with similar characteristics in lack of having available the ones from the Torqeedo motors.

The last term in Eq. (14) corresponds to the singularity avoidance term, where $\varepsilon > 0$ is required to avoid division by zero in the quotient. The singularity condition is attained when $\det(\boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{T}^T(\boldsymbol{\alpha})) = 0$, which also translates on matrix $\boldsymbol{T}(\boldsymbol{\alpha})$ not having full rank. This means that the generalized vector force $\boldsymbol{\tau}_c$ in Fig. 2 is restricted to a subspace of $\mathbb{R}^3$ such that the system become underactuated while losing the ability to produce a desired generalized force in any direction in $\mathbb{R}^3$ [12]. Moreover, a large value of parameter ϱ implies high maneuverability at the cost of increased power consumption [6].

IV. Reinforcement Learning Based on Artificial Neural Networks

A. The concept of Reinforcement Learning

Reinforcement learning (RL) [15] is a process by which an 'agent' (in this case, the autonomous USV) learns to earn rewards through trial-and-error interactions with its 'environment'. This trial-and-error nature allows the agent to operate with virtually complete autonomy, which is one of the distinguishing features of RL compared to traditional control techniques (such as a feedback controller). Autonomous decision-making allows RL agents to adapt effectively to unforeseen circumstances, a characteristic that is vital when operating in unfamiliar surroundings, or in continually changing environmental conditions.

The 'environment' in our case is represented by the set of ordinary differential equations (2) that describe the 3-degree-of-freedom motion of the USV. Before making a control-decision, the agent considers its 'state', i.e., its present circumstances in its given environment. In case of the USV, the state-vector may include variables such as range and bearing with respect to certain inertial fixed points, the speed and rotation of the craft, and its motor thrust values and azimuth angles, to name a few.

The learning-agent serves as a controller for the USV, and its purpose is to determine the best control-inputs ('actions') that will allow the USV to collect maximum reward over the long-term. In the present work, the agent selects 4 action-values that control the maneuvering of the USV, namely, 2 thrust values for the port and starboard motors, and their respective azimuth angles. The state-vector consists of 10 scalar quantities: the range and bearing from the start of the target path (r, θ); the yaw angle in the inertial reference frame (ψ); the longitudinal, lateral, and rotational velocities in the inertial frame ($\dot{x}, \dot{y}, \dot{\psi}$); the port and starboard thrust values (T_{port}, $T_{starboard}$); and the port and starboard motor azimuth angles (δ_{port}, $\delta_{starboard}$). This collection of 10 variables allows us to comprehensively describe the 'state' of the USV with respect to its environment (e.g., a testing arena, or the open ocean).

As mentioned previously, the action-vector consists of 4 values corresponding to the thrust and azimuthal angles of the two motors. These values are received by the agent from the Neural Network. We note that these variables are also part of the state-vector.

The 'reward' is one of the most important components of an RL algorithm, since it conveys the high-level objective that the agent must try to accomplish. We note that when designing a problem using RL, shaping the reward function is one of the most important ways in which we can influence the overall behavior of the agent, especially since we play no direct role in managing the low-level decision-making process of the agent.

B. The Bellman equation

The optimality of the actions selected by the agent depends on the Bellman equation, which aims to maximize the total cumulative reward received throughout an experiment:

$$V^{\pi^*}(s_t) = \max_{a_t} \left(r(s_t, a_t) + \gamma \sum_{s_{t+1}} P(s_{t+1}|a_t, s_t)\, V^{\pi^*}(s_{t+1}) \right)$$

Here, π^* represents the optimal 'policy' that encodes the behavior of the agent, and γ represents the discount factor, which emphasizes long-term rewards over short-term benefits. The emphasis on long-term reward is an important distinguishing feature of RL, since it allows the agent the freedom to choose actions that may be detrimental in the immediate future, but which may prove to be the best possible choice for maximizing the reward received over the long term. In the present work, the value $\gamma=0.99$ is used for all training runs.

The optimal policy π^* is determined through trial-and-error interactions of the agent with the environment. During training, the agent observes the state of the environment s_t at every new turn, and performs an action a_t. $P(s_{t+1}|a_t, s_t)$ denotes the probability that this particular action will cause the agent to a transition to a new state s_{t+1}. The action thus influences both the transition to the next state and the reward received r_{t+1}. The agent's goal is to learn the optimal control policy $a_t = \pi^*(s_t)$ that maximizes the action value $V^{\pi^*}(s_t)$, which in turn represents the sum of discounted future rewards.

C. Training using Artificial Neural Networks

One of the primary tasks in RL is to determine the best policy π^* that guides the agent's actions. In simple terms, the policy can be thought of as a multivariate function, where the state-variables act as inputs, and the actions act as outputs. Thus, the task of training an agent involves determining a suitable functional-approximator for the policy, which relates the input state-values to the most appropriate action-values. When considering relatively simple problems, trained policies may take the form of 'tables', where the best action value for every possible combination of state-variables is tabulated in a grid. However, such tables quickly become unwieldy when considering a large number of action- and state-variables. In the current scenario, where we are concerned with 10 distinct state-variables, the resulting grid would be 10-dimensional for each of the 4 action-variables.

To mitigate the prohibitive computational cost associated with the tabulation approach, we rely on an Artificial Neural Network (ANN) to act as a functional-approximator for the optimal policy. We use an ANN consisting of an input layer, an output layer, and 3 hidden-layers, with each hidden-layer being comprised of 128 individual nodes. The specific RL algorithm used for training the ANN is referred to as 'V-RACER' [16].

During training, the USV (learning agent) is initialized with randomized initial positions, velocities, and yaw angle within a small region from the starting point of the target path. The *x* and *y* positions are selected from a Normal distribution with its mean centered on the path's starting point, and standard deviation equal to the length of the craft. The two initial velocity components are also drawn from a Normal distribution with the same characteristics. The initial yaw angle with respect to the target path is initialized from a Normal distribution with a standard deviation of 10 degrees from the target path. The randomized initialization of the state-variables is vital to prevent 'overfitting' of the ANN, and to ensure that the learned policy is general enough to be effective when the agent starts with different initial conditions, or when it encounters unforeseen environmental perturbations.

During training, both the state vector and the corresponding reward are communicated to the neural network at pre-determined time-intervals (0.1s in our case). The agent, in turn, receives the 4 action values from the Neural Network, which allow it proceed forward in time by integrating the equations of motion (2), and end up in a new state. The agent then evaluates the reward it received by performing this action. This reward function is a combination of various objectives that we want the craft to attain. For instance, in our case, the craft's objective is to reach the end of the path, while following the prescribed trajectory and maintaining the appropriate heading. To achieve the first goal, we specify the path's end as the termination goal, and we award the agent with a positive reward (+10 units) if it manages to reach within a radius of 0.1L (10% of the craft's length) from the end point.

We note that while the positive reward allotted at the path's end will eventually cause the agent to adopt a behavior that attracts it towards the target, it might take a significant number of trial-and-error training runs by the agent before it discovers this relatively small beneficial region. To speed up training, we specify another stepwise reward that increases as the agent moves closer to the target: $[L_{path} - r_{end}]$, which includes the total

length of the path and the agent's Euclidean distance from the target end-point, projected along the path.

With this reward specification, the agent will learn to move towards the end point without any regard to its heading, speed, or cross-track error. To shape the desired behavior with respect to these quantities, we also allocate negative stepwise reward to the learning agent. For instance, to minimize the cross-track error, we specify a reward equal to *[-100Δy]*, which is proportional to the relevant error. Eventually, the total stepwise reward allocated at each turn is taken to be the linear sum of various individual stepwise reward quantities. We note that in order to speed up training, we also limit the extent to which the agent can deviate from the target path. If the agent exceeds a cross-track error (lateral distance) of 10 meters from the target path, the training-simulation terminates and the agent receives a large terminal negative reward (-2e5). This large punishment discourages the agent from exploring regions that are far away from the target path, and results in shorter training durations.

In future studies, we will also consider step-wise rewards related to power drawn by the motors, in order to optimize the energy-efficiency of path-following operations.

V. Simulations And Results

A. Path-Following Control Results

Simulations were carried out using Matlab's Simulink graphical module at a sample frequency of 100 Hz in order to resemble the actual output frequency of a sensor data acquisition system. The objective was to simulate the entire control process as close as possible to the real implementation, taking into account electro-mechanical limitations of the thrusters, as explained in the control allocation section. The desired path consists of two orthogonal straight lines forming a flipped L-shape, which allows for assessment of both control methods over periods of steady-state and transitional conditions. Below is a description of the simulation conditions and parameters, as well as observations regarding the results depicted in Fig. 4.

- Simulation conditions and parameters:
  - USV initial pose: $x = 6;\ y = 0;\ \psi = 90°$;
  - No environmental forces;
  - Cruise-speed: $u = 2\frac{m}{s}$ (desired surge speed);
  - Thrust: $T_{max} = 350\ N;\ T_{min} = -250\ N$ (for both thrusters);
  - Power: $P_{max} = 1120\ W$; $P_{min} = 0\ W$ (for both thrusters);
  - Steering: $\alpha_{min} = -45\ °;\ \alpha_{max} = 45\ °$ (for both thrusters);
- The results shown in Fig. 4 were obtained under no environmental forces; however, the controller was also tested in presence of disturbances with similar results, apart from larger steady-state errors. When no disturbances are included, the steady-state error is almost null in all the control variables, as shown in the figure.

  The first row in Fig. 4 shows the results regarding the control variables. The first plot shows the actual path (in red) with respect to the desired path (dashed-blue) while the vehicle tries to maintain the cruising speed of $2\ m/s$, as shown in the second plot. One can notice how this control approach does an acceptable job controlling the surge speed and the heading, not only when the vehicle reaches a steady-state condition, but also in the transitional region, near the vertex of the reference path, where it effectively reacts to the corresponding sudden changes.

  The second row in Fig. 4 depicts the response of both thrusters (port and starboard) with respect to time, in terms of azimuth angles (first plot), thrust (second plot) and propulsive power (third plot), where Eq. (19) was used. It can be noticed how the control allocation approach provides smooth rates of change within predefined saturation limits for azimuths and thrust.
- Errors corresponding to the control variables, surge speed, cross-track error and heading – are condensed in TABLE I. The metric used is the root mean squared error (RMSE) and its cumulative value.

TABLE I. RMSE Of Control Variables

Variable	**RMSE**
Surge Speed [*m/s*]	0.46951
Cross-Track Error [*m*]	1.70576
Heading [*rad*]	0.2492 (14.3°)
Cumulative RMSE	2.42448

- One of the aspects to compare against the performance of the *RL* -based control approach is the total power consumption associated to the generated thrust, a.k.a. propulsive power. TABLE II shows these values for each thruster, as well as the cumulative value.

TABLE II. Propulsive Power

Thruster	**Power [W]**
Port motor	439721
Starboard motor	428210
Cumulative Power	867931

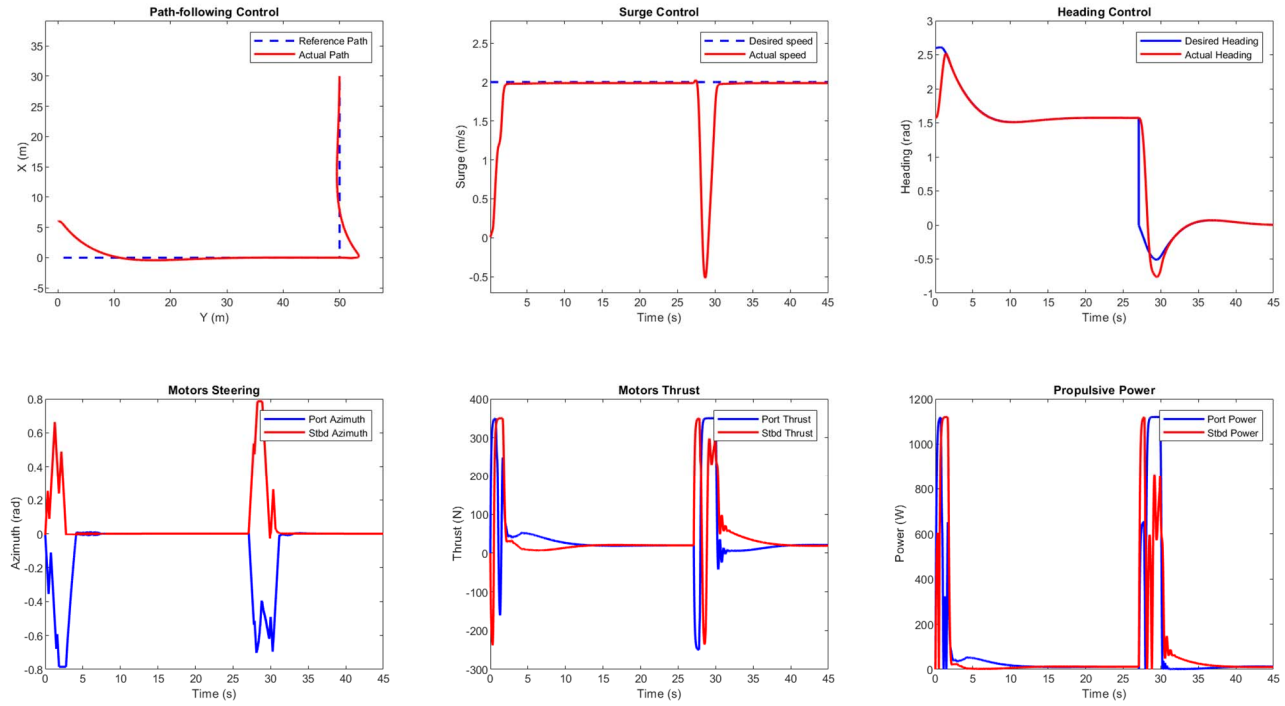

Fig. 4. Results obtained using Path-following PD-controller. Top row: control variables results. Bottom row: response of the thrusters over time

B. Reinforcement Learning based Control

For assessing the effectiveness of the RL-based controller, we conducted learning-simulations using the equations of motion provided in Eqs. 1 through 6. As described in Section IV, the USV (RL agent) is not provided with explicit directives on what actions it must take to achieve its objective. Instead, the agent conducts a large number of trial-and-error runs, over which it attempts to attain the specified high-level goal. The intended goal is conveyed to the agent through a reward function, which in our case consists of the following combination of variables:

$$r(s) = -(\,100\,\mathrm{cte} + 100\,|\alpha_k - \psi| + 40\left|2 - \sqrt{\dot{x}^2 + \dot{y}^2}\right| - 50\,\left(1 - \exp\left(-\frac{d_{k+1}}{40}\right)\right)) \tag{22}$$

Here, d_{k+1}represents the parallel distance of the USV from the target point $\vec{p}_{k+1}$, i.e., its displacement from point $\vec{p}_{k+1}$, projected along the vector $(\vec{p}_{k+1} - \vec{p}_k)$. The form of the reward function described in Eq. 22 is fairly complex, and it attempts to drive the agent towards the desired behavior with regard to the desired cross track error, heading, and speed. The negative sign in front of the reward function discourages the agent (negative reinforcement) from taking actions that may cause it to deviate from the specified objectives. The first term attempts to minimize the cross-track error, by penalizing non-zero values of the *cte*. The second term is related to heading control. The third term, which depends on the inertial velocity components, punishes the agent anytime its speed deviates from the specified value of 2.0 m/s. The final exponential term acts as positive reinforcement, encouraging the agent to move towards the target point $\vec{p}_{k+1}$.

We note that rewards are allotted to the agent in three distinct scenarios. Every time the agent takes an action and ends up in a new state, it receives the 'stepwise' reward described by Eq. 22. Furthermore, the agent is also allotted 'terminal' reward at the end of each simulation, which may be positive or negative depending on the outcome of the agent's actions. In our case, the agent is allotted a positive reward of +10 units upon reaching the target $\vec{p}_{k+1}$. In a different outcome, it receives a negative terminal reward of -200,000 units if its cross-track error exceeds 10 m, or if its speed exceeds 10 m/s at any point. The large negative reward provides a strong incentive for the agent to stay within a bounded region and within a certain speed limit, in order to accelerate learning by suppressing actions that would cause the USV to stray too far away from the target path. We remark that varying the magnitudes of these rewards can have a notable influence on the overall behavior of the agent. For instance, if the positive terminal reward specified is too large, the agent may start ignoring the stepwise rewards in an attempt to get to the final target point as quickly as possible (so as to avoid the accumulation of negative stepwise rewards). While this may seem like a decent strategy at first glance, the agent might end up failing all but one of the specified goals. More specifically, in our case, the USV might put on a great spurt of speed at the beginning of the simulation (which may not be realistic) and approach the target point. While this would satisfy the goal of reaching the target point, our primary objective of traversing the specified path would be forfeit.

To be able to make adaptive decisions, the agent must undergo a training procedure which is accomplished with the help of Artificial Neural Networks in the present study. During training, the agent periodically receives state and reward information from the environment (the equations of motion

being solved in time), and responds with the best action to be taken. In our case, the agent communicates with the environment every 0.1 seconds, whereas the time-step size for advancing the equations of motion is 1e-3 seconds. This implies that the environment proceeds for 100 time steps, before communicating with the RL controller. The state-vector for the agent consists of the 10 state-variables described previously in Section IV. The action-vector consists of 4 numbers: the port and starboard motor thrusts, and their azimuth angles. We limit the possible thrust values that can be selected by the RL-controller to be in the range [-250, 350] N, whereas the motor azimuth angles are limited to the range $[-45°, 45°]$. For the 100 time-steps when there is no communication between the environment and the controller (agent), the thrusts and azimuth angles are transitioned from their old values to the new controller-assigned values. The change in the azimuth angle is done linearly, with the maximum rate magnitude being limited to 45°/second. The change in thrust is accomplished using a critically-damped step-response with time-constant $\tau = 0.1$s:

$$T_{transient} = T_{old} + (T_{new} - T_{old})\left(1 - \exp\left(-\frac{\Delta t}{0.1}\right)\right) \quad (23)$$

We remark that environmental forces were not considered for training the RL agent. The impact of such environmental disturbances will be considered in future studies.

For training the autonomous agent, we use the V-RACER reinforcement learning algorithm, which is implemented in an open source C++ code [15]. The feedforward Neural Network used consists of an input layer, an output layer, and three hidden layers comprised of 128 nodes each. For each training simulation, the agent is initialized at a random position within a radius of 1.89 m from the starting point $\vec{p}_k$. The initial velocities are also picked at random from a Gaussian distribution with mean 0 and standard deviation 1.89 m/s. Only positive values for the initial surge speed are allowed, which is ensured by taking the absolute value of the relevant random number drawn. The initial heading is selected as a Gaussian random number with mean equal to the heading-angle of the path and standard deviation equal to 10°. The equations of motion are integrated forward in time using the 4th order Runge-Kutta method.

As the trial-and-error simulations proceed, the unknown coefficients of the Artificial Neural Network are continually updated in accordance with the V-RACER algorithm. Training continues until the cumulative reward no longer changes with additional simulations, and the ANN has reached a steady converged state. For our case, training converges after approximately 15,000 simulations, at which point the agent's policy may be considered to be optimal. The policy is then 'evaluated' to obtain the autonomous behavior of the agent, i.e., the actions of the USV are based on the thrust and azimuth values provided to it by the ANN. Oftentimes, the results of policy-evaluation do not exhibit the intended behavior of the agent, and various forms of the reward function and Neural Network parameters (hyperparameters) must be tested before a satisfactory solution is reached.

The resulting behavior of the trained agent is shown in Fig. 5, which includes the trajectory of the USV as well as the time evolution of various output- and control-variables. We re-iterate that these results do not include the influence of environmental forces. The resulting path of the autonomous USV corresponds very well with the desired path, as can be observed in the panel on the top left. The USV is also able to maintain the specified surge speed of 2 m/s, with deviations observed at initial startup, and close to the 25 second mark, where the USV must maneuver to make a 90° turn towards port. Heading-control also shows good performance with respect to the desired outcome, with

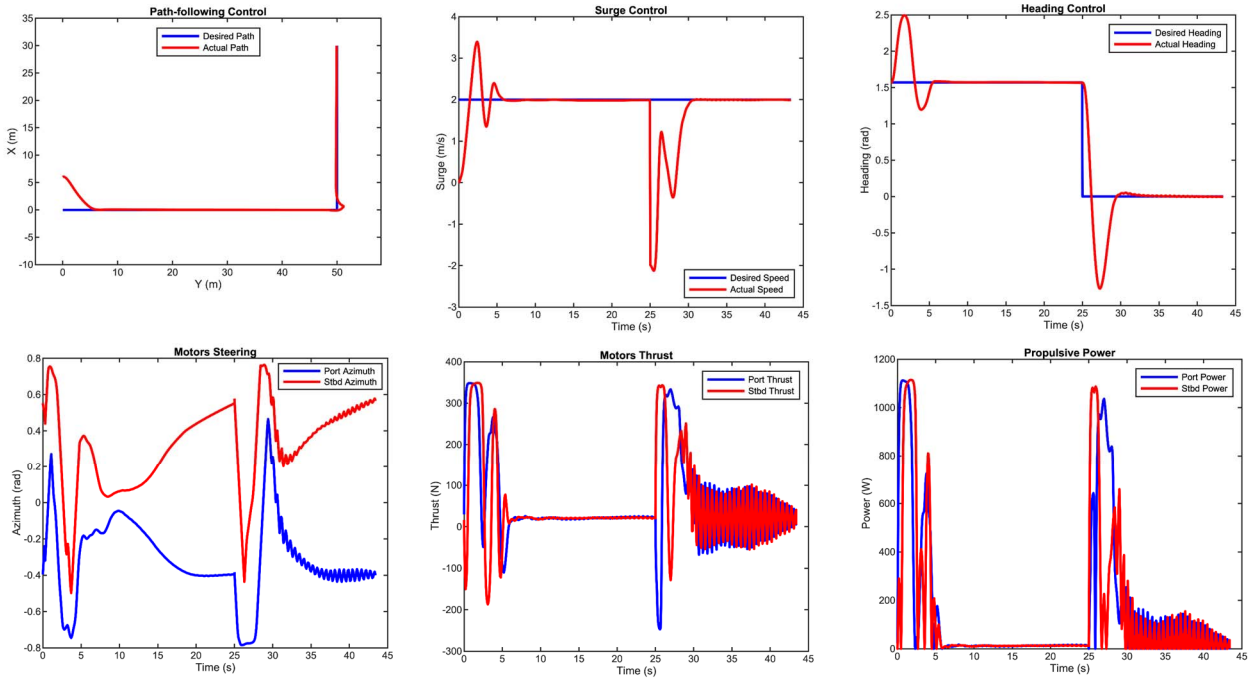

Fig. 5. Results obtained using the RL-based controller. Top row: control variables results. Bottom row: response of the thrusters over time

deviations evident at startup and the turning point. We observe that the control-input values (motor azimuth and thrust – bottom left and center panels) fluctuate continuously in time, which is indicative of the RL-controller making minute control adjustments throughout the simulation. Looking closely at the motor thrust graph, we observe an initial positive peak for the port motor, and a corresponding negative peak for the starboard motor, which indicates that the USV is relying on differential thrust to attain the initial heading change that is necessary to bring it closer to the desired path from its starting location. We again observe peaks in the thrust at 25s, but with opposing signs this time, which assists the USV in making the necessary turn to port. The propulsive power shown in the bottom right panel displays corresponding peaks at 0s and 25s.

Errors corresponding to the control variables, surge speed, cross-track error and heading are provided in TABLE III. TABLE IV shows the values associated with the propulsive power for each thruster, as well as the cumulative value.

TABLE III. RMSE OF CONTROL VARIABLES

Variable	**RMSE**
Surge Speed [*m/s*]	0.8052
Cross-Track Error [*m*]	1.3527
Heading [*rad*]	0.3449 (19.8°)
Cumulative RMSE	2.5028

TABLE IV. PROPULSIVE POWER

Thruster	**Power [W]**
Port motor	6,681,200
Starboard motor	6,083,800
Cumulative Power	12,765,000

Our results confirm conclusively that Reinforcement Learning may serve as a robust option to implement autonomous control for the USV, especially in situations where continually modulating control inputs (motor thrust and azimuth angles) may prove to be beneficial.

VI. CONCLUSIONS

The results obtained in Fig. 4 and Fig. 5 reveal the potential of Reinforcement Learning-based algorithms for controlling the motion of USVs, particularly the WAM-V 16 USV, along a path (path-following-control). This is reflected in the RMSE value for the cross-track error over the entire path as shown in TABLE III, compared to the PD-controller in TABLE I, where this value is larger by around 40 cm. This precise tracking of the desired path, however, comes with a cost in terms of power consumption, as shown in TABLE IV, where the propulsive power (associated with the generated thrust) is almost 15 times greater for the RL-based method compared to the traditional control approach in TABLE II. This was somewhat expected given that the PD-controller approach was optimized for power consumption in the control allocation stage, while the objective for the RL controller was centered around maneuverability, instead. Power consumption is one of the most important aspects in the area of USV motion control, provided that more often than not, these vehicles are equipped with electric thrusters, powered by batteries, constituting a limiting factor in terms of endurance in actual implementations.

It is also important to remark that, in general, the PD-controller performed better in controlling the desired surge speed and heading, not only in terms of the RMSE value, but also in terms of the transient response of these control variables, involving lesser oscillations in path following (Fig. 4). This however may change if the same enclosure-based method was incorporated to obtain the desired heading in the training stage of the RL approach, which avoids extreme changes associated with this value.

Another problem, which is common to all kinds of controllers, but clearly evidenced in the RL control method, is the chatter associated with the actuators in the vehicle, especially in the last portion of the thrust vs. time response in Fig. 5, where high-frequency components can be seen. This behavior is in general undesirable since it implies high control action in real electro-mechanical devices and may excite unmodeled high-frequency dynamics [3]. Nonetheless, these high frequency motions can be naturally damped in pulse-width modulated electric motors, since the control input is voltage rather than, for example, a force, so that the chatter frequency is beyond the frequency range of the relevant unmodeled dynamics [3].

REFERENCES

[1] E. I. Sarda, I. R. Bertaska, A. Qu and K. D. Ellenrieder, "Development of a USV station-keeping controller," in *OCEANS 2015*, Genova, 2015.

[2] M. W. Spong, Robot Modeling and Control, John Wiley, 2012.

[3] J.-J. E. Slotine and W. Li, Applied Nonlinear Control, Upper Saddle River, New Jersey: Prentice-Hall, Inc., 1991.

[4] W. B. Klinger, I. R. Bertaska and K. D. von Ellenrieder, "Experimental testing of an adaptive controller for USVs with uncertain displacement and drag," in *Oceans*, St. John's, 2014.

[5] H. Ashrafiuon, K. R. Muske, L. C. McNinch and R. A. Soltan, "Sliding-Mode Tracking Control of Surface Vessels," *IEEE Transactions on Industrial Electronics,* vol. 55, no. 11, pp. 4004-4012, 2008.

[6] T. I. Fossen, Handbook of Marine Craft Hydrodynamics and Motion Control, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom: John Wiley & Sons, Ltd., 2011.

[7] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction, Cambridge,MA, USA: The MIT press, 1998.

[8] G. Reddy, J. Wong-Ng, A. Celani, T. J. Sejnowski and M. Vergassola, "Glider soaring via reinforcement learning in the field," *Nature,* vol. 562, no. 7726, pp. 236-239, 2018.

[9] S. Verma, G. Novati and P. Koumoutsakos, "Efficient collective swimming by harnessing vortices through deep reinforcement learning," *Proceedings of the National Academy of Sciences,* vol. 115, no. 23, pp. 5849-5854, 2018.

[10] M. A. Diddams, Control of an Unmanned Surface Vehicle Using An Environmental Disturbance Observer, Boca Raton, Florida, USA: MSc. Thesis, Florida Atlantic University, 2018.

[11] "WAM-V," Marine Advanced Robotics, Inc, 2005-2019. [Online]. Available: http://www.wam-v.com/. [Accessed 12 September 2019].

[12] T. I. Fossen, R. Skjetne and M. Breivik, "Line-of-Sight PathFollowing of Underactuated Marine Craft," in *IFAC Conference on Maneuvering and Control of Marine Craft*, Girona, Spain, 2003b.

[13] T. A. Johansen, T. I. Fossen and S. P. Berge, "Constrained Nonlinear Control Allocation With Singularity Avoidance Using Sequential Quadratic Programming," *IEEE Transactions On Control Systems Technology,* vol. 12, no. 1, pp. 211-216, 2004.

[14] C. De Wit, "Optimal Thrust Allocation Methods for Dynamic Positioning of Ships," MSc Thesis, Delft University of Technology, Delft, the Netherlands, 2009.

[15] "Torqeedo," Torqeedo, 2019. [Online]. Available: https://www.torqeedo.com/us/en-us/technology-and-environment/performance-and-efficiency.html. [Accessed 12 September 2019].

[16] G. Novati and P. Koumoutsakos, "Remember and Forget for Experience Replay," in *36th International Conference on Machine Learning*, Long Beach, California, USA, 2019.