USV Path-Following Control Based On Deep Reinforcement Learning and Adaptive Control

Alejandro Gonzalez-Garcia
Tecnologico de Monterrey
School of Science and Engineering
Monterrey, Mexico
a00818249@itesm.mx

Herman Castañeda
Tecnologico de Monterrey
School of Science and Engineering
Monterrey, Mexico
hermancc@tec.mx

Leonardo Garrido
Tecnologico de Monterrey
School of Science and Engineering
Monterrey, Mexico
leonardo.garrido@tec.mx

Global Oceans 2020: Singapore - U.S. Gulf Coast | 978-1-7281-5446-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/IEEECONF38699.2020.9389360

***Abstract*—This paper presents a guidance and control scheme for an unmanned surface vehicle. The approach combines a deep reinforcement learning based guidance law that can learn the dynamics of vessel with an adaptive sliding mode controller to achieve path-following. The guidance implements a deep deterministic policy gradient algorithm to obtain the desired heading command, whereas the adaptive control drives the heading and surge speed. The proposed guidance has self-learning ability based on evaluative feedback, which does not require any prior knowledge of the dynamic system, and the controller exhibits robustness against bounded uncertainties and perturbations, control gain non-overestimation, and chattering reduction. Simulation results show that the proposed guidance and control law achieves fast convergence and small overshoot, and improved performance when compared against line-of-sight based guidance laws.**

***Index Terms*—Unmanned surface vehicle, deep reinforcement learning, adaptive sliding mode control, path-following, marine robotics.**

I. Introduction

Unmanned Surface Vehicles (USVs) are marine systems capable of operating without any on-board crew, and full-autonomy is required to reduce human intervention and error in different activities considered “dirty, dull, and dangerous” as it is explained in [1], [2]. One of the common problems in the USV autonomy literature is path-following control, defined in [3] as following a predefined path independent of time. Path-following controllers are composed by a cascaded system of a guidance law and a low-level controller.

There are several guidance laws. However, one of the most implemented is to follow a Line-Of-Sight (LOS) based approach, as in [3]–[9]. The standard LOS is further addressed in [3], [4]. A more recent Integral LOS (ILOS) is presented in [5], which improves the steady-state error performance of the LOS guidance law. Another kind of guidance law used in the USV literature is the Vector Field Guidance (VFG) method, as used in [11].

On the other hand, Artificial Intelligence (AI) methods, such as Machine Learning (ML) based approaches have been recently proposed for USV control problems. A ML technique which learns directly from interaction with its environment is Reinforcement Learning (RL), which can be combined with the layered Neural Networks (NNs) concepts from Deep Learning (DL) to create the field of Deep Reinforcement Learning (DRL). Although most RL and DRL methods are discrete, some algorithms such as Deep Deterministic Policy Gradient (DDPG) [10] can deal with continuous action-spaces, as the path-following control problem demands. RL and DRL techniques have been applied both for low-level control and path-following control scenarios (see for more details [11]–[19]). DRL has been implemented in path-following control to directly actuate the system, either with a guidance law input [11], or acting as both guidance law and heading controller [12], [13]. Hence, guidance law strategies that can benefit from the DRL characteristics have not been proposed. Likewise, as DRL strategies have only been validated by performing training in simulation, they are susceptible to model uncertainties. If the Deep Neural Networks (DNNs) are only trained offline, DRL strategies may benefit from robust control schemes such as Adaptive Sliding Mode Control (ASMC) addressed in [20], [21]. The ASMC is a control strategy which presents robustness against bounded uncertainties and perturbations, control gain non-overestimation, and chattering reduction.

The main contribution of this paper is a path-following control strategy based on a DDPG-based guidance law and an ASMC to control surge speed and heading. A reward function is proposed to minimize the cross-tracking error and chattering in the desired heading commands. Simulation results with the VTec S-III USV model [22] showcase the feasibility of the proposed approach and the improvement in performance in comparison to LOS and ILOS based guidance laws.

The organization of this paper is as follows: Section II describes the VTec S-III USV model and the ASMC surge speed and heading control strategy. Section III addresses the DDPG-based guidance law. The Section IV presents the training and simulation results. Finally, conclusions are drawn.

II. USV Modeling and Control Design

In this section, the mathematical derivation of a 3 degree of freedom model is introduced. Moreover, the design of the low-level control based on a class of adaptive sliding mode control is also addressed.

A. USV Dynamic Model

The VTec S-III is a USV platform developed at Tecnologico de Monterrey, and its dynamic model was developed using

978-1-7281-5446-6/20/$31.00 ©2020 IEEE

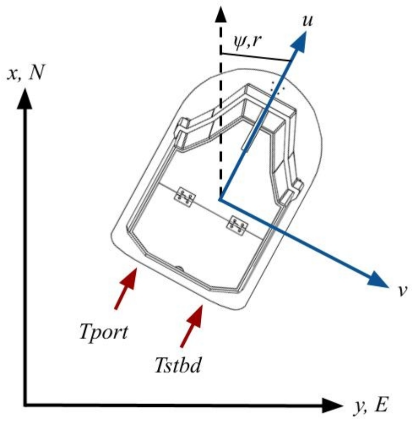

Fig. 1. NED and body-fixed reference frames.

the robot-like model for marine craft [3] in [22]. Vector $\boldsymbol{\eta} = [x, y, \psi]^T$ describes the (x,y) Cartesian position in the North-East-Down (NED) reference frame, and the yaw angle ψ. Likewise, $\boldsymbol{v} = [u, v, r]^T$ depicts the body fixed linear velocities (u,v), and the yaw rate r. Therefore, the dynamics of the USV are represented by:

$$\boldsymbol{\tau} = \boldsymbol{M}\dot{\boldsymbol{v}} + \boldsymbol{C}(\boldsymbol{v})\boldsymbol{v} + \boldsymbol{D}(\boldsymbol{v})\boldsymbol{v} \quad (1)$$

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\boldsymbol{\eta})\boldsymbol{v} \quad (2)$$

where (2) describes the kinematics, with $\boldsymbol{R}(\boldsymbol{\eta})$ being a transformation matrix between body-fixed and NED reference frames, which is given by:

$$\boldsymbol{R}(\boldsymbol{\eta}) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$\boldsymbol{M}$ is the sum of a rigid body mass matrix and an added mass matrix:

$$\boldsymbol{M} = \begin{bmatrix} m - X_{\dot{u}} & 0 & -my_G \\ 0 & m - Y_{\dot{v}} & mx_G - Y_{\dot{r}} \\ -my_G & mx_G - N_{\dot{v}} & I_z - N_{\dot{r}}, \end{bmatrix} \quad (4)$$

$\boldsymbol{C}(\boldsymbol{v})$ is a Coriolis matrix, the sum of a rigid body matrix and an added mass matrix:

$$\boldsymbol{C}(\boldsymbol{v}) = \begin{bmatrix} 0 & 0 & -m(x_G r + v) \\ 0 & 0 & -m(y_G r - u) \\ m(x_G r + v) & m(y_G r - u) & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 2(Y_{\dot{v}}v + (\frac{Y_{\dot{r}} + N_{\dot{v}}}{2})r) \\ 0 & 0 & -X_{\dot{u}}u \\ 2(-Y_{\dot{v}}v - (\frac{Y_{\dot{r}} + N_{\dot{v}}}{2})r) & X_{\dot{u}}u & 0 \end{bmatrix} \quad (5)$$

TABLE I
VTEC S-III PHYSICAL PARAMETERS

Parameter	Value
Length overall (L)	1.01 [m]
Draft (T)	0.09 [m]
Beam overall	0.75 [m]
Centerline-to-centerline side hull separation (B)	0.41 [m]
Individual hull beam (B_{hull})	0.27 [m]
Mass (m)	30 [kg]
Longitudinal center of gravity	0.45 [m]
Moment of inertia (I_z)	4.1 $[kgm^2]$

TABLE II
HYDRODYNAMIC COEFFICIENTS

Coefficient	Non-dimensional Factor	Dimensional Term
X_u		25, ($u > 1.2$) 64.55
Y_v	0.5	$f(Y,v)$
Y_r	3	$-\pi\rho\sqrt{(u^2+v^2)}T^2L$
N_v	0.06	$-\pi\rho\sqrt{(u^2+v^2)}T^2L$
N_r	0.02	$-\pi\rho\sqrt{(u^2+v^2)}T^2L^2$
$X_{\dot{u}}$		-2.25
$Y_{\dot{v}}$		-23.13
$Y_{\dot{r}}$		-1.31
$N_{\dot{v}}$		-16.41
$N_{\dot{r}}$		-2.79
$X_{u\|u\|}$		0, ($u > 1.2$) -70.92
$Y_{v\|v\|}$		-99.99
$Y_{v\|r\|}$		-5.49
$Y_{r\|v\|}$		-5.49
$Y_{r\|r\|}$		-8.8
$N_{v\|v\|}$		-5.49
$N_{v\|r\|}$		-8.8
$N_{r\|v\|}$		-8.8
$N_{r\|r\|}$		-3.49

$$f(Y,v) = -40\rho|v|\left[1.1 + 0.0045\frac{L}{T} - 0.1\frac{B_{hull}}{T} + 0.016\left(\frac{B_{hull}}{T}\right)^2\right]\left(\frac{\pi T L}{2}\right)$$

$\boldsymbol{D}(\boldsymbol{v})$ is the sum of linear and nonlinear drag matrices:

$$\boldsymbol{D}(\boldsymbol{v}) = -\begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix} - \begin{bmatrix} X_{u|u|}|u| & 0 & 0 \\ 0 & Y_{v|v|}|v| + Y_{v|r|}|r| & Y_{r|v|}|v| + Y_{r|r|}|r| \\ 0 & N_{v|v|}|v| + N_{v|r|}|r| & N_{r|v|}|v| + N_{r|r|}|r| \end{bmatrix} \quad (6)$$

$\boldsymbol{\tau}$ is a vector of forces created by two rear thrusters with differential thrust, expressed by:

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} T_{port} + \Phi T_{stbd} \\ 0 \\ (T_{port} - \Phi T_{stbd})B/2 \end{bmatrix} \quad (7)$$

where $\Phi = 0.78$ denotes a mechanical constrain of the VTec S-III [23]. Finally, Table I [22] contains the physical parameters of the USV, and Table II contains the hydrodynamic coefficients and equations, as shown in [22], [23]. In-depth explanation of the model parameters can be found in [22].

B. USV Control Design

An adaptive sliding mode control strategy [20], [21] is proposed to drive the surge speed and heading dynamics of the VTec S-III. In order to design the controllers, the full model (1)-(2) can first be simplified into separate and reduced dynamics following the assumptions reported in [24]–[26]. Then, the surge dynamics are expressed by:

$$\dot{\zeta} = f(\zeta, t) + g(\zeta)U \tag{8}$$

where $\zeta = u$ is the surge speed, and

$$f(\zeta, t) = \frac{1}{m - X_{\dot{u}}}\left[(m - Y_{\dot{v}})vr + X_{u|u|}u|u| + X_u u\right] \tag{9}$$

$$g(\zeta) = \frac{1}{m - X_{\dot{u}}} \tag{10}$$

$$U = \tau_x \tag{11}$$

Then, the surge speed error, is defined as follows:

$$e_u = (u_d - u) \tag{12}$$

where u_d is the expected surge velocity. Now, a sliding surface s_u is defined as:

$$s_u = e_u + \lambda_u \int e_u \tag{13}$$

where $\lambda_u > 0$. Then, by taking the time derivative of (13), we can define the closed-loop dynamics:

$$\begin{aligned} \dot{s}_u &= \dot{e}_u + \lambda_u e_u \\ &= \dot{u}_d - \dot{u} + \lambda_u(u_d - u) \\ &= \dot{u}_d - (f(\zeta, t) + g(\zeta)U) + \lambda_u(u_d - u) \end{aligned} \tag{14}$$

Hence, the feedback controller $U = \tau_x$ can be addressed by:

$$\tau_x = \frac{1}{g(\zeta)}\left[-f(\zeta, t) + \dot{u}_d + \lambda_u(u_d - u) - \delta_u\right] \tag{15}$$

where δ_u is taken by the adaptive sliding mode strategy:

$$\delta_u = -K_i(t)|s_u|^{(1/2)}\operatorname{sign}(s_u) - K_2 s_u \tag{16}$$

with adaptive gain $K_i(t)$ given by:

$$\dot{K}_i(t) = \begin{cases} k_r \operatorname{sign}(|s_i| - \mu), & \text{if } K_i > k_{min} \\ k_{min}, & \text{if } K_i \leq k_{min} \end{cases} \tag{17}$$

and fixed gain K_2. This control technique exhibits robustness to bounded uncertainties and perturbations, reduced chattering effect, and non overestimation of the control gain. The mechanics of the controller act as follows: A working domain μ detects the loss of sliding mode, adjusting K_i. k_{min} is a gain that ensures no zero gain, and k_r denotes the adaptation rate. On the other hand, the heading dynamics are represented by:

$$\begin{aligned} \dot{\zeta} &= \zeta_2 \\ \dot{\zeta}_2 &= f(\zeta, t) + g(\zeta)U \end{aligned} \tag{18}$$

where $\zeta = \psi$ is the yaw angle, $\zeta_2 = r$ is the yaw rate, and

$$f(\zeta, t) = \frac{1}{I_z - N_{\dot{r}}}\left[(-X_{\dot{u}} + Y_{\dot{v}})vu + N_{r|r|}r|r| + N_r r\right] \tag{19}$$

$$g(\zeta) = \frac{1}{I_z - N_{\dot{r}}} \tag{20}$$

$$U = \tau_\psi \tag{21}$$

Next, the heading error as:

$$e_\psi = (\psi_d - \psi) \tag{22}$$

with ψ_d being the desired heading. Then, a sliding surface is now defined as:

$$s_\psi = \dot{e}_\psi + \lambda_\psi e_\psi \tag{23}$$

with $\lambda_\psi > 0$ and $\dot{\psi} = r$. Now, the time derivative of s_ψ gives:

$$\begin{aligned} \dot{s}_\psi &= \ddot{e}_\psi + \lambda_\psi \dot{e}_\psi \\ &= (\dot{r}_d - \dot{r}) + \lambda_\psi(r_d - r) \\ &= \dot{r}_d - (f(\zeta, t) + g(\zeta)U) + \lambda_\psi(r_d - r) \end{aligned} \tag{24}$$

Thus, the following feedback controller $U = \tau_\psi$ is presented as:

$$\tau_\psi = \frac{1}{g(\zeta)}\left[-f(\zeta, t) + \dot{r}_d + \lambda_\psi(r_d - r) - \delta_\psi\right] \tag{25}$$

where δ_ψ follows the ASMC strategy:

$$\delta_\psi = -K_i(t)|s_\psi|^{(1/2)}\operatorname{sign}(s_\psi) - K_2 s_\psi \tag{26}$$

Notice that the system (1)-(2) is underactuated, where sway v control is neglected. This is due to the assumption that the proposed USV is mechanically stable by design, *i.e.*, the effect from lateral motion is bounded and small [2].

III. Deep Deterministic Policy Gradient Based Guidance Law

In this section, the path-following control problem is stated. Then, the concepts of reinforcement learning and the deep deterministic policy gradient algorithm are described to further design and implement the proposed guidance law.

A. Path-Following Control Problem

The path-following control problem is defined by [3] as following a predefined path independent of time under no temporal restrictions. The USV is assumed to maintain a constant speed, hence desired heading commands are given to the low-level heading controller to achieve the control objective.

A straight-line path can be expressed by two sets of coordinates $\boldsymbol{p}_k = [x_k, y_k]^T$ and $\boldsymbol{p}_{k+1} = [x_{k+1}, y_{k+1}]^T$ in the NED frame. Then, a local reference frame called Path Parallel (PP) is defined at $\boldsymbol{p}_k$, with angle α_k defined as:

$$\alpha_k = atan2(y_{k+1} - y_k, x_{k+1} - x_k) \tag{27}$$

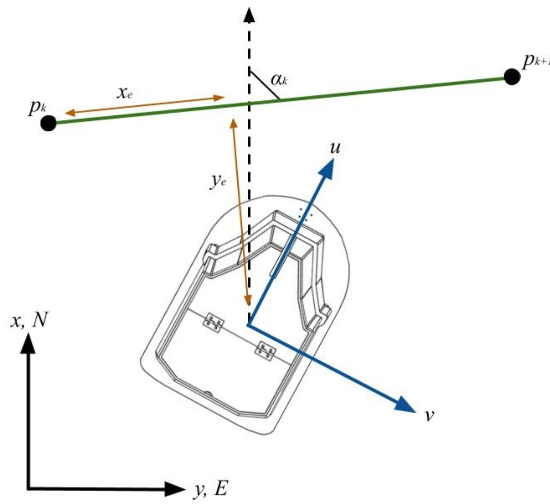

Fig. 2. Path-following control problem geometry.

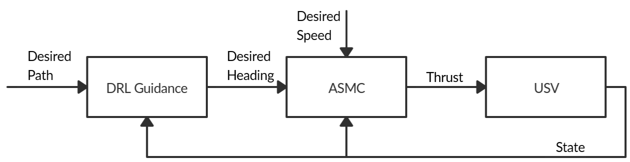

Fig. 3. Closed-loop diagram.

which can be used to obtain the error vector $\boldsymbol{p}_e = [x_e, y_e]^T$, where x_e is the along-tracking error, and y_e is the cross-tracking error computed by:

$$y_e = -(x - x_k)\sin\alpha_k + (y - y_k)\cos\alpha_k \tag{28}$$

The control objective is to converge into the desired path. Thus, the cross-tracking error is desired to converge to zero. Fig. 2 illustrates the path-following control problem, and Fig. 3 represents the closed-loop dynamics including the USV dynamics, the ASMC low-level controller, and the proposed guidance law.

B. Reinforcement Learning

Reinforcement learning is a branch of ML that focuses on goal-directed learning from interaction between an agent and its environment [27]. It is assumed that the environment behavior can be modeled by a Markov Decision Process (MDP). A MDP has a state space S, an action space A, an initial state distribution $p(s_1)$, a transition model $p(s_{t+1}|s_t, a_t)$, and a reward function $r(s_t, a_t)$. At each timestep t, the agent receives an observation s_t, takes an action a_t and obtains a reward r_t. An agent's behavior is described by a policy π, which maps states to a probability distribution over the actions. The goal is to find an optimal policy π^* that maximizes the return from a state, which is the accumulated discounted reward $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$, where $\gamma \in [0, 1]$ is known as the discount factor. Action-value functions are widely used in reinforcement learning algorithms [10]. These functions express the expected discounted reward after taking an action a_t in state s_t, and follows the policy π, as:

$$Q(s_t, a_t) = \mathbb{E}_{s_{i>=t}, a_{i>=t} \sim \pi}[R_t | s_t, a_t] \tag{29}$$

The combination of RL with DL establishes the field of DRL [28]. One class of DRL algorithm is the actor-critic algorithm, which approximates both the policy and action-value functions with a function approximator such as a deep neural network (DNN).

C. Deep Deterministic Policy Gradient

The DDPG algorithm is an actor-critic DRL method developed by [10], and its capability of "robustly solving challenging problems across a variety of domains with continuous action space" [10] makes it suitable for the guidance problem. The policy function $\pi(s|\theta_a)$ and the action-value function $Q(s, a|\theta_c)$ are DNNs, where θ_a and θ_c are the DNN parameters. To update both functions, stochastic gradient descent is executed on batches $\mathbb{B}$ of transitions (s_i, a_i, r_i, s_{i+1}) following the rules:

$$\theta_c \leftarrow \theta_c - \alpha_c \frac{1}{N} \sum_{i \in \mathbb{B}} \nabla_{\theta_c} (y_i - Q(s_i, a_i|\theta_c))^2) \tag{30}$$

$$\theta_a \leftarrow \theta_a - \alpha_a \frac{1}{N} \sum_{i \in \mathbb{B}} \nabla_{a_i} Q(s_i, a_i|\theta_c) \nabla_{\theta_a} \pi(s_i|\theta_a) \tag{31}$$

where α_c and α_a are learning rates, and y_i is the action-value estimate, or Temporal Difference (TD) target, obtained from:

$$y_i = r_i + \gamma Q'(s_{i+1}, \pi'(s_{i+1}|\theta_{a'})|\theta_{c'}) \tag{32}$$

Here, $\theta_{c'}$ and $\theta_{a'}$ are parameters from two target networks. The target networks are introduced in DDPG to stabilize training [10], by slowly learning the parameters using the following equations:

$$\theta_{c'} = (1 - \tau)\theta_{c'} + \tau\theta_c \tag{33}$$

$$\theta_{a'} = (1 - \tau)\theta_{a'} + \tau\theta_a \tag{34}$$

where τ is the target network update rate. Fig. 4 depicts an overview of the DDPG architecture and the MDP, whereas Algorithm 1 describes the DDPG algorithm [10].

D. Implementation

To implement DDPG for the path-following control problem, the state is first defined as $S = [u, r, \psi_{\alpha_k}, y_e, \dot{y}_e, a_p]$, where $\psi_{\alpha_k} = \psi - \alpha_k$ is the heading error relative to the PP angle, and a_p is the previous action taken. Likewise, the action space is $A = [a]$. The desired heading is then defined as $\psi_d = \psi + a$, hence $a = e_\psi$, and $a \in [-\pi/2, \pi/2]$.

The reward function is composed by three partial rewards:

$$r_y = \max \begin{cases} e^{-k_y |y_e|} \\ e^{-k_y y_e^2} \end{cases} \tag{35}$$

$$r_a = w_a \tanh(-c_a \dot{a}^2) \tag{36}$$

$$r_\psi = -e^{k_\psi(|\psi_{\alpha_k}| - \pi)} \tag{37}$$

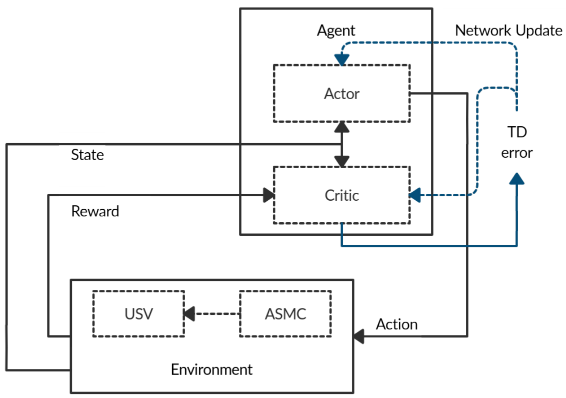

Fig. 4. DDPG architecture.

Algorithm 1: DDPG algorithm.

Randomly initialize critic network $Q(s, a|\theta_c)$ and actor network $\pi(s|\theta_a)$ with weights θ_c and θ_a
Initialize target networks Q' and π' with weights $\theta_{c'} \leftarrow \theta_c$ and $\theta_{a'} \leftarrow \theta_a$
Initialize replay buffer R
for *episode = 1, M* **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = *1, T* **do**
    Select action $a_t = \pi(s_t|\theta_a) + \mathcal{N}_t$ according to the current policy and exploration noise
    Take action a_t, observe reward r_t and observe new state s_{t+1}
    Save transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions from R
    Set y_i with Equation (32)
    Update the critic by minimizing the loss using Equation (30)
    Update the actor policy using Equation (31)
    Update the target networks using Equations (33) and (34)

where $k_y = 0.5$, $w_a = 0.2$, $k_\psi = 5.72$, and

$$c_a = 1/(\frac{a_{max} - a_{min}}{\Delta t})^2 \tag{38}$$

Equation (35) minimizes the cross-tracking error by using a combination of gaussian and exponential functions. Equation (36) minimizes large variations between taken actions, to achieve a smooth desired heading signal. Equation (37) maintains the USV heading toward the correct direction of the path. Fig. 5 illustrates the partial reward functions. Then, the weighted reward function is given by:

$$r = \begin{cases} r_y + r_a, & \text{if } |\psi_{\alpha_k}| < \pi/2 \\ r_\psi, & \text{if } |\psi_{\alpha_k}| \geq \pi/2 \end{cases} \tag{39}$$

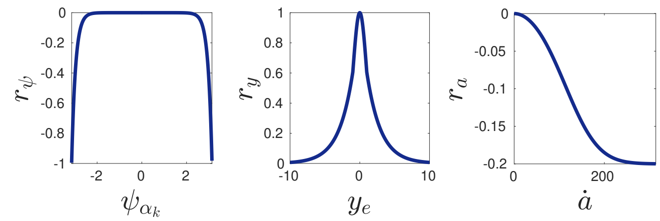

Fig. 5. Reward signals.

Moreover, the DNN architecture and hyper-parameters were selected to be similar to the expressed in the original paper [10], as the actor network inputs the state, has two hidden layers of 400 and 300 neurons respectively, and outputs the action. The activation functions are Rectified Linear Units (ReLU) for the hidden layers, and a hyperbolic tangent activation function for the output. The critic network inputs the state, then has a hidden layer of 400 neurons, next adds a hidden layer of 300 neurons which also inputs the action taken, then has a third hidden layer, composed of 300 neurons, and outputs the action-value. All of the hidden layers have ReLU activation functions, except the output, which has a linear activation function.

IV. Training and Results

In this section, the deep neural network training environment and process is explained. Furthermore, simulation results are presented.

A. Deep Neural Network Training

To train the DNNs, the gradient descent-based Adam optimizer [29] was used to learn the DNN parameters with learning rates of 10^{-4} and 10^{-3} for the actor and critic DNNs respectively, and $\gamma = 0.99$. The critic DNN also included an L_2 weight decay of 10^{-2}. The soft target DNN updates included $\tau = 0.001$. Hidden layers weights were initialized using a Glorot uniform function, whereas the output layers weights were initialized with a random uniform distribution $[-3 \times 10^{-3}, 3 \times 10^{-3}]$. The size of the training minibatches was of 64, using a replay buffer of size 10^6. The exploration noise $\mathcal{N}$ used was an Ornstein-Uhlenbeck process [30] with $\theta = 0.15$ and $\sigma = 0.2$.

The training data consisted of simulating different straight-line paths, while starting the USV in random positions and orientations, and $\boldsymbol{v}_0 = [0, 0, 0]$. Each episode consisted of 400 training steps, with a timestep of 0.05 seconds, which simulates 20 seconds running the guidance law at 20 Hz. However, the ASMC was running in the simulation at a higher frequency of 100 Hz, which is used in practical implementation [20]. Thus, for every training step, the ASMC runs 5 times before the next observation. This decision was made because, in practice, several guidance systems need to run at lower frequencies, particularly when perception systems, such as computer vision with cameras or LiDAR, are involved. Therefore, this architecture can be expanded into different problems such as collision avoidance. Furthermore, each episode had a

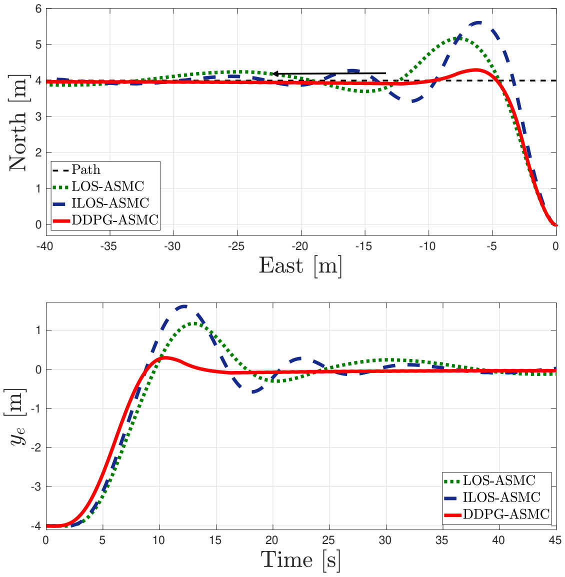

Fig. 6. **Scenario I.** Path and USV trajectory (top), cross-tracking error (bottom).

random maximum desired speed $u_{d,max} \in [0.4, 1.4]$ so that the USV could learn from different dynamic scenarios. As the USV could start in any orientation, a speed selection function was implemented to assist the USV in 180° turns, given by:

$$u_d = (u_{d,max} - u_{d,min}) \frac{1}{1 + e^{\kappa(|e_\psi|\chi - \Delta_o)}} + u_{d,min} \quad (40)$$

where $u_{d,min} = 0.3$, $\kappa = 10$, $\chi = 2/\pi$, and $\Delta_o = 0.5$. Additionally, the ASMC gains were $k_{r,u} = 0.1$, $K_{2,u} = 0.02$, $k_{min,u} = 0.05$, $\lambda_u = 0.001$, $\mu_u = 0.05$, $k_{r,\psi} = 0.2$, $K_{2,\psi} = 0.1$, $k_{min,\psi} = 0.2$, $\lambda_\psi = 1$ and $\mu_\psi = 0.1$. Finally, training was stopped after 1500 episodes.

B. Simulation Results

1) Scenario I: The first scenario required the USV to follow a desired straight-line path at a target speed of 1 m/s, and an initial heading parallel to the path. The DDPG guidance law was compared against a standard LOS guidance law [3] with look-ahead distance $\Delta = 2$, and an ILOS guidance law [5] with look-ahead distance $\Delta = 1$ and integral gain $\sigma = 0.1$, both using the ASMC as low-level speed and heading controller.

The simulation results of the first scenario are shown in Fig. 6, where DDPG achieves a better performance at following the path. The cross-tracking error plot shows the DDPG guidance has less overshoot and is faster at reaching steady-state. The Mean Square Error (MSE) of DDPG is lower than the LOS MSE and the ILOS MSE, as seen in the performance index in Table III.

2) Scenario II: The second scenario required the USV to follow a zig-zag path with a target speed of 1 m/s, and an initial heading perpendicular to the path. As in the first scenario, the second scenario shows a lower DDPG cross-tracking error overall as seen in Fig. 7. Here, it is also noticeable that when

TABLE III
SCENARIO I. PERFORMANCE INDEX.

Guidance Law	y_e **MSE**
DDPG-ASMC	0.0025 [m]
LOS-ASMC	0.0281 [m]
ILOS-ASMC	0.0264 [m]

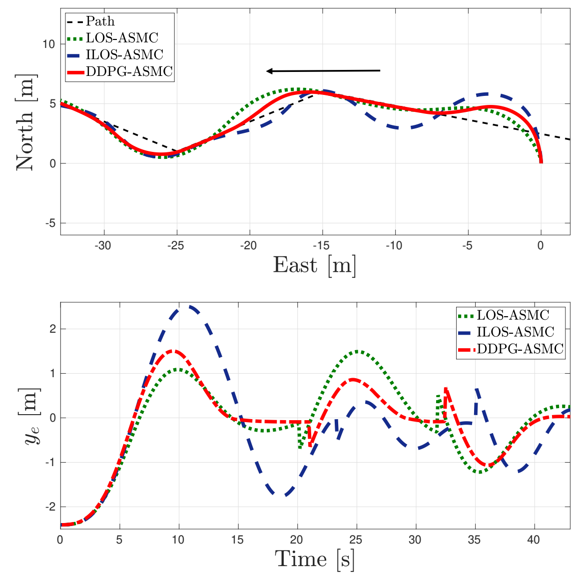

Fig. 7. **Scenario II.** Path and USV trajectory (top), cross-tracking error (bottom).

starting closer to the path and facing toward it, more overshoot is present on the DDPG trajectory than on the LOS trajectory, differing from the first scenario. However, the DDPG guidance still has a faster stabilization than the LOS and ILOS.

V. CONCLUSIONS

A guidance and control scheme for an unmanned surface vehicle has been addressed. The method included a deep deterministic police gradient algorithm based guidance law that learned the boat dynamics without knowledge of the model nor the controller, where a proper heading reference was provided. Furthermore, an adaptive sliding mode strategy was designed to drive the heading and speed surge in spite of bounded uncertainties, achieving the USV path-following. The performance and feasibility of the proposed approach were demonstrated via two scenarios in simulation such as keeping the north in a straightforward way and a zig-zag like trajectory, where our methodology exhibited fast convergence and small overshoot when opposed to a standard LOS and more recent ILOS guidance laws.

ACKNOWLEDGMENTS

This work was supported by the student group VantTec, as well as their sponsors: Hacsys, VectorNav, Google, ifm efector, Uber ATG, RoboNation, Velodyne Lidar, NVIDIA, Skysset, Akky and Greenzie. Finally, the authors acknowledge the support from the university, Tecnologico de Monterrey.

REFERENCES

[1] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, “Unmanned surface vehicles: An overview of developments and challenges,” Annual Reviews in Control, vol. 41, pp. 71-93, 2016.

[2] M. Breivik, “Topics in guided motion control of marine vehicles“, PhD. Thesis, Norwegian University of Science and Technology NTNU, June 2010.

[3] T. I. Fossen, “Handbook of Marine Craft Hydrodynamics and Motion Control,” John Wiley & Sons, Ltd., 2011.

[4] F. Papoulias, “Bifurcation analysis of line of sight vehicle guidance using sliding modes,“ International Journal of Bifurcation and Chaos, vol. 1, no. 4, pp. 849-865, 1991.

[5] W. Caharija et al., “Integral Line-of-Sight Guidance and Control of Underactuated Marine Vehicles: Theory, Simulations, and Experiments,” IEEE Trans. Control Syst. Technol., vol. 24, no. 5, pp. 1623–1642, 2016.

[6] T. I. Fossen, K. Y. Pettersen, and R. Galeazzi, “Line-of-sight path following for Dubins paths with adaptive sideslip compensation of drift forces,” IEEETrans. Control Syst. Technol., vol. 23, no. 2, pp. 820–827, 2014.

[7] T. I. Fossen, K. Y. Pettersen, and R. Galeazzi, “Line-of-sight path following for Dubins paths with adaptive sideslip compensation of drift forces,” IEEE Trans. Control Syst. Technol., vol. 23, no. 2, pp. 820–827, Mar. 2015.

[8] L. Liu, D. Wang, Z. Peng, et al., “Predictor-based LOS guidance law for path following of underactuated marine surface vehicles with sideslip compensation,” Ocean Engineering, vol. 124, pp. 340-348, Sep. 2016.

[9] A.M. Lekkas and T.I. Fossen, “A Time-Varying look-ahead Distance Guidance Law for Path Following,” 9th IFAC Conference on Manoeuvring and Control of Marine Craft, 2012.

[10] T.P. Lillicrap et al., “Continuous control with deep reinforcement learning,“ https://arxiv.org/pdf/1509.02971.pdf, 2016.

[11] J. Woo, C. Yu and N. Kim, “Deep reinforcement learning-based controller for path following of an unmanned surface vehicle,“ Ocean Engineering, vol. 183, p. 155-166, July 2019.

[12] A. Martinsen and A. Lekkas, “Straight-Path Following for Underactuated Marine Vessels using Deep Reinforcement Learning,“ OCEANS 2018 MTS/IEEE Charleston, 2018.

[13] A. Martinsen and A. Lekkas, “Curved Path Following with Deep Reinforcement Learning: Results from Three Vessel Models,“ IFAC-PapersOnLine, vol. 51, pp. 329-334, 2018.

[14] Y. Wang et al., “Unmanned surface vehicle course tracking control based on neural network and deep deterministic policy gradient algorithm,“ OCEANS 2018 MTS/IEEE Kobe Techno-Ocean (OTO), 2018.

[15] M. De Paula et al., “Trajectory tracking algorithm for autonomous vehicles using adaptive reinforcement learning,“ OCEANS 2015 - MTS/IEEE Washington, 2015.

[16] L. Zhang et al., “Neural-network-based reinforcement learning control for path following of underactuated ships,“ 2016 35th Chinese Control Conference (CCC), 2016.

[17] A. Martinsen et al., “Reinforcement Learning-Based Tracking Control of USVs in Varying Operational Conditions,“ Frontiers in Robotics and AI, 2020.

[18] E. Meyer, H. Robinson, A. Rasheed and O. San, “Taming an Autonomous Surface Vehicle for Path Following and Collision Avoidance Using Deep Reinforcement Learning,” in IEEE Access, vol. 8, pp. 41466-41481, 2020, doi: 10.1109/ACCESS.2020.2976586.

[19] A. Gonzalez-Garcia et al., “Control of an Unmanned Surface Vehicle Based on Adaptive Dynamic Programming and Deep Reinforcement Learning,“ Proceedings of the 2020 4th International Conference on Deep Learning Technologies (ICDLT 2020), 2020.

[20] A. Gonzalez-Garcia and H. Castañeda, “Guidance and Control Based on Adaptive Sliding Mode Strategy for a USV Subject to Uncertainties,“ unpublished.

[21] H. Castanẽda, J. Rodriguez and J.L. Gordillo, “Continuous and smooth differentiator based on adaptive sliding mode control for a quad-rotor MAV,” Asian Journal of Control, Nov. 2019, DOI:10.1002/asjc.2249.

[22] A. Gonzalez-Garcia and H. Castañeda, “Modeling, Identification and Control of an Unmanned Surface Vehicle,“ AUVSI XPONENTIAL 2019: All Things Unmanned, 2019.

[23] A. Gonzalez-Garcia and H. Castañeda, “Control of a Double Thruster Twin-Hull Unmanned Surface Vehicle: Experimental Results,“ XXI Congreso Mexicano de Robótica 2019, 2019.

[24] N. Wang et al., “Finite-Time Observer Based Guidance and Control of Underactuated Surface Vehicles With Unknown Sideslip Angles and Disturbances,” IEEE Access, vol. 6, pp. 14059 - 14070, Jan. 2018.

[25] J. Nie and X. Lin, “Robust Nonlinear Path Following Control of Underactuated MSV With Time-Varying Sideslip Compensation in the Presence of Actuator Saturation and Error Constraint,” IEEE Access, vol. 6, pp. 71906 - 71917, Nov. 2018.

[26] J. Miao et al., “Spatial curvilinear path following control of underactuated AUV with multiple uncertainties,” ISA Transactions, vol. 67, pp. 107-130, Mar. 2017.

[27] R.S. Sutton and A. Barto, “ Reinforcement Learning: An Introduction,” MIT Press, 1998.

[28] K. Arulkumaran et al., “Deep Reinforcement Learning: A Brief Survey,” IEEE Signal Processing Magazine, 2017.

[29] D.P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,“ 2014.

[30] G.E. Uhlenbeck and L.S. Ornstein, “On the theory of the brownian motion,“ Physical review, 1930.