Ocean Engineering 183 (2019) 155–166

Contents lists available at ScienceDirect

Ocean Engineering

journal homepage: www.elsevier.com/locate/oceaneng

Deep reinforcement learning-based controller for path following of an unmanned surface vehicle

Joohyun Woo [a], Chanwoo Yu [b], Nakwan Kim [c,*]

[a] Institute of Engineering Research, Seoul National University, 1, Gwanak-ro, Gwanak-gu, Seoul, 08826, Republic of Korea
[b] The 6th R&D Institute — 3rd Directorate, Agency for Defense Development, Dong-eup, Uichang-gu, Jinhae, Changwon, 51698, Republic of Korea
[c] Research Institute of Marine Systems Engineering, Seoul National University, 1, Gwanak-ro, Gwanak-gu, Seoul, 08826, Republic of Korea

ARTICLE INFO

Keywords:
Deep reinforcement learning
Path following
Unmanned surface vehicle
Learning-based control
Artificial intelligence

ABSTRACT

In this paper, a deep reinforcement learning (DRL)-based controller for path following of an unmanned surface vehicle (USV) is proposed. The proposed controller can self-develop a vehicle's path following capability by interacting with the nearby environment. A deep deterministic policy gradient (DDPG) algorithm, which is an actor-critic-based reinforcement learning algorithm, was adapted to capture the USV's experience during the path-following trials. A Markov decision process model, which includes the state, action, and reward formulation, specially designed for the USV path-following problem is suggested. The control policy was trained with repeated trials of path-following simulation. The proposed method's path-following and self-learning capabilities were validated through USV simulation and a free-running test of the full-scale USV.

1. Introduction

Recently, unmanned surface vehicles (USVs) have been actively adopted in both military and commercial areas. In terms of safety, USVs can replace humans in tasks performed in dangerous and extreme environments such as mine countermeasures (Bertram, 2008), anti-submarine warfare (Corfield and Young, 2006), and reconnaissance (Kucik, 2004). In terms of economic efficiency, USVs can replace humans in performing repetitive and tedious missions such as resource exploration (Majohr and Buch, 2006) or oceanic sample acquisition (Naeem et al., 2006). Although there are different types of USVs designed for different tasks, their common feature is their path-following capability, which is one of the most fundamental capabilities that any USV must possess. For this, a USV must have a properly designed path-following controller that is effective and robust.

Various control algorithms have been developed by many researchers to implement path following in unmanned marine vehicles. These include the proportional–integral–derivative control (Lekkas and Fossen, 2012; Bibuli et al., 2009), sliding mode control (Meng et al., 2012), backstepping control (Sonnenburg and Woolsey, 2013), fuzzy control (Zhu et al., 2016; Garus and Zak, 2010), and model predictive control (Naeem et al., 2006). In recent years, there have been many attempts to deal with uncertainty in the control domain, such as disturbance or dynamic modeling uncertainty. An adaptive control algorithm (Shin et al., 2017; Mu et al., 2017) has been developed in an attempt to consider the uncertainty in the vehicle's dynamic model. In Lu et al. (2018), minimal learning parameter (MLP) and the disturbance observer (DOB) techniques are adopted to deal with USV formation control. Similarly, a robust neural algorithm was adopted to deal with uncertainty in Zhang et al. (2018), while the USV is performing path-following control in presence of multiple obstacles. Although a number of USV path-following controllers have been proposed, most of the approaches have limitations in terms of dependency on prior knowledge of dynamic modeling and handling of modeling uncertainty. To overcome such limitations, we adopted recently devised machine learning techniques to develop a USV controller. The proposed deep reinforcement learning (DRL)-based USV controller has self-learning capability, and, thus, it does not require any prior knowledge of USV dynamics to tune the controller. Similarly, because the learning process directly uses experience data, consisting of the control input and the corresponding dynamic response of the vehicle, the controller can implicitly consider modeling the uncertainty or the effect of environmental disturbances as well.

In recent years, there have been several attempts to implement machine learning techniques in the field of unmanned maritime vehicles (Bertaska, 2016; Bertaska and von Ellenrieder, 2018; Cheng and Zhang, 2018; Magalhães et al., 2018; Carreras et al., 2003; Zhang et al., 2014; De Paula and Acosta, 2015; Yoo and Kim, 2016; Woo, 2018; Cui et al., 2017). Bertaska (2016) used a traditional Q-learning approach to develop a supervisory switching controller for USVs. In this work, three

* Corresponding author.
E-mail address: nwkim@snu.ac.kr (N. Kim).

https://doi.org/10.1016/j.oceaneng.2019.04.099
Received 10 November 2018; Received in revised form 28 April 2019; Accepted 29 April 2019
Available online 18 July 2019

basis controllers were specially designed for transiting, station keeping, and reversing maneuver, respectively. Then, a Q-learning algorithm was used to intelligently switch the basis controller of the USV according to the current situation and maneuver of the USV. A full-scale USV experiment was conducted to validate the proposed method. Cheng and Zhang (2018) adopted a DRL technique for obstacle avoidance of an underactuated unmanned marine vehicle. In this work, convolutional layers were used to capture the obstacle's information and a reward was specially designed for obstacle avoidance tasks. Magalhães et al. (2018) devised a conventional Q-learning-based reinforcement learning technique to develop a biomimetic underwater vehicle controller. In their work, the Q-learning-based controller mimicked the stroke style of a fish (with two lateral fins and a tail). The control policy was trained by using a biomimetic vehicle simulator, and the learned model was transferred to the real vehicle for validation. Woo and Kim (2016) adopted the deep reinforcement learning in the field of collision avoidance of an unmanned surface vehicle. By expressing the encounter situation using a grid map based visual information, an unmanned surface vehicle can deal with avoidance decision making in complex encounter situation. Similar to Magalhães et al. (2018), a policy model was trained in simulation environment, and additional validation experiment was conducted.

Some of the contributions of the proposed method to the field of marine vehicle control are as follows. First, this work was the first implementation of continuous action-space reinforcement learning in the control of an unmanned marine vehicle. There exist some works that used reinforcement learning for control purposes (Magalhães et al., 2018; Carreras et al., 2003); however, they defined action space as discrete actions with only a few action candidates. The small number of action candidates hinders the precise control of a vehicle or often results in the chattering phenomenon. Second, we applied the DRL-based controller to a full-scale USV control and conducted a number of real-world experiments for validation. In most of the works related to reinforcement learning on a marine vehicle, the training and the validation domain were restricted to simulation. In contrast, we extended the domain into a real-world domain to show its practical applicability.

The rest of the paper is organized as follows. In Section 2, the USV dynamics and path-following system used in this work are presented. In Section 3, the theoretical background and problem formulation of the reinforcement learning-based controller are given. In Section 4, the validation of the proposed method through full-scale USV path-following experiments is discussed, as well as the test results and analysis conducted. Finally, in Section 5, the main conclusions and additional discussions are presented.

2. USV path following

According to the theoretical background on reinforcement learning, a learning-based controller is developed through a repeated training process. During the training, the controller will perform an action that is then evaluated on the basis of the response of the environment to the conducted action. Because of these characteristics, a learning-based controller requires an "environment" to interact with, to be trained. In this work, we conducted USV path-following simulations to train the controller.

Fig. 1 shows a schematic diagram of the USV path-following system for the training simulation. As illustrated, a dynamic system should be implemented to describe the vehicle's dynamic behavior and a guidance block is adopted to determine the desired course angle. The rest of this section provides detailed information about the components of the USV path-following system.

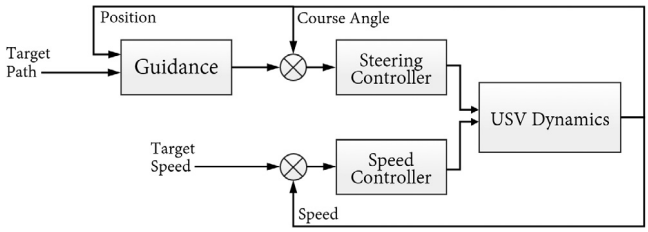

Fig. 1. Block diagram of the USV path-following system.

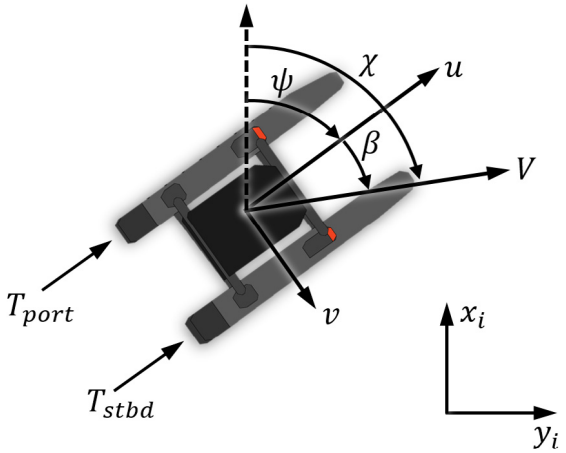

Fig. 2. Coordinate system and differential thrust of the USV.

2.1. USV dynamic model

In this work, we selected a 16-ft wave adaptive modular vessel (WAM-V) platform as the target USV type. A WAM-V platform is a catamaran-shaped platform with two inflatable pontoons hulls with a wave shock observer (described in Fig. 2). The vehicle was equipped with two electric thrusters on each hull, and the thruster vector angle was fixed, as illustrated in Fig. 2. Because of this characteristic, the vehicle generated yaw motion by making RPM differences in the two thrusters (known as differential thruster type). Woo et al. (2018) recently conducted a dynamic system identification on an identical platform. In his work, several dynamic model including simplified dynamics and LSTM based deep neural network model was identified. In this work, we adopted the identification result of the simplified dynamic model represented in Woo et al. (2018) as a simulation model. Once the simulation-based training process was finished, the trained controller was implemented on the WAM-V platform and path-following experiments were conducted to validate its performance.

Since a horizontal planar motion of the USV is the primary concern in a path-following problem, we mainly focused on 3-degrees-of freedom (3DOF) USV motions (surge, sway, and yaw motion). For the notation, the one proposed by Fossen (2002) was used. The coordinate system used in this work is illustrated in Fig. 2, where x_i and y_i are the north and the east directional position, respectively, of the USV in the inertial frame, whereas u, v, and V are the surge, sway, and total speed, respectively, of the vehicle in a body-fixed frame. The heading angle of the USV is defined as ψ, whereas the course angle and the side slip angle are represented by χ and β, respectively. By definition, the side slip angle β can be calculated by the equation $\beta = asin(\frac{v}{V})$ and the course angle can be described by the sum of ψ and β, as in the expression $\chi = \psi + \beta$.

The USV kinematic model can be described as in (1), where $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ are defined as the velocity vector and the position vector, respectively.

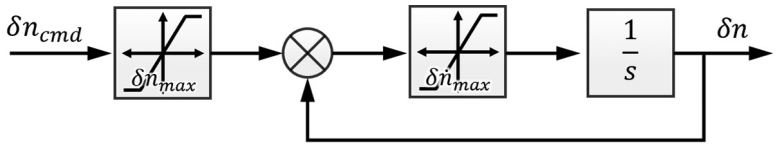

Fig. 3. Schematic diagram of the actuator dynamics of the WAM-V USV platform.

$\boldsymbol{R}(\boldsymbol{\eta})$ is a rotation matrix from a body-fixed frame to an inertial frame.

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

$$\boldsymbol{\nu} = (u, v, r)^T \tag{2}$$

$$\boldsymbol{\eta} = (x_i, y_i, \psi)^T \tag{3}$$

$$\boldsymbol{R}(\boldsymbol{\eta}) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

The 3DOF horizontal planar dynamic model of the USV can be described by the following equation:

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{f}, \tag{5}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{C}(\boldsymbol{\nu})$ is the Coriolis and centripetal matrix, $\boldsymbol{D}(\boldsymbol{\nu})$ is the damping matrix, and $\boldsymbol{f}$ is the control forces and moment. Since the target USV was a differential thruster type, the control forces and moment $\boldsymbol{f}$ can be described as follows:

$$\boldsymbol{f} = \begin{bmatrix} \tau_X \\ \tau_Y \\ \tau_N \end{bmatrix} = \begin{bmatrix} T_{port} + T_{stbd} \\ 0 \\ (T_{port} - T_{stbd}) \cdot \frac{B}{2} \end{bmatrix} \tag{6}$$

In the above equation, T_{port} and T_{stbd} represent the thrust force of the port side and the starboard side thruster, respectively, and B refers to the beam of the target USV. We implemented an actuator model based on model of Woo et al. (2018); thus, the thrust force can be directly calculated from the thruster's RPM δn as in (7).

$$T = 3.54 \times 10^{-5}\delta n^2 + 0.084\delta n - 3.798, \tag{7}$$

where the RPM δn of either the port or the starboard side is determined by the steering command δn_d and the speed command δn_m, which are inversely calculated from (8) and (9), respectively. In (8), the steering command δn_d is defined as the RPM difference between δn_{port} and δ_{stbd}, normalized by the maximum RPM value δn_{max}. Similarly, the speed command is defined as a normalized mean value of both RPMs as in (9).

$$\delta n_d = (\delta n_{port} - \delta n_{stbd})/(2\delta n_{max}) \tag{8}$$

$$\delta n_m = (\delta n_{port} + \delta n_{stbd})/(2\delta n_{max}) \tag{9}$$

According to the actuator dynamics presented in Woo et al. (2018), the absolute value of the RPM δn and the rate of the RPM $\delta \dot{n}$ are saturated as δn_{max} and $\delta \dot{n}_{max}$, respectively, as shown in Fig. 3.

Among the various dynamic models suggested in Woo et al. (2018), we adopted the linearized maneuvering dynamics as the model for the simulation. Owing to the simplicity of the selected model, the calculation time of the model is fast enough for running millions of simulation steps from the repetitive training process in a limited time, while maintaining an appropriate level of dynamics prediction accuracy. For the linearized maneuvering dynamics, the speed dynamic model is described as in (10) and the steering dynamics can be represented as in (11):

$$\dot{u} = a_u u + b_u u\tau_X + b_{u_{bias}} \tag{10}$$

$$\begin{bmatrix} \dot{v} \\ \dot{r} \end{bmatrix} = \boldsymbol{A}\begin{bmatrix} v \\ r \end{bmatrix} + \boldsymbol{B}\tau_N + \boldsymbol{B_{bias}}, \tag{11}$$

where the matrix $\boldsymbol{A}$ is a 2-by-2 system matrix, the matrix $\boldsymbol{B}$ is a 2-by-1 control matrix, and $\boldsymbol{B_{bias}}$ is a 2-by-1 matrix for inclusion of the bias term. According to the system identification result in Woo et al. (2018), the unknown parameters in (10) and (11) can be described as in (12) and (13):

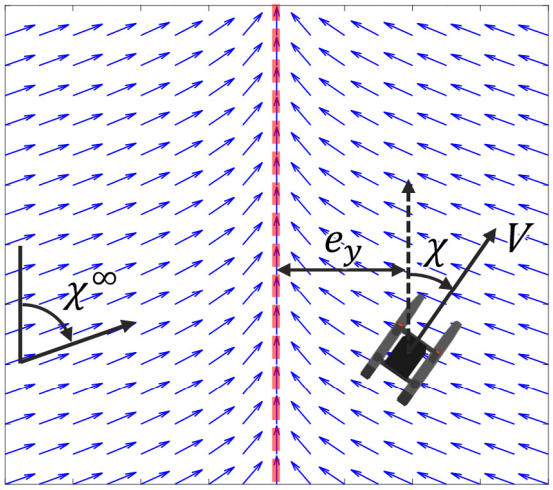

Fig. 4. Vector field guidance for USV linear path following.

$$\dot{u} = -1.3191u + 0.0028u\tau_X + 0.6836 \tag{12}$$

$$\begin{bmatrix} \dot{v} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0.0161 & -0.0052 \\ 8.2861 & -0.9860 \end{bmatrix}\begin{bmatrix} v \\ r \end{bmatrix} + \begin{bmatrix} 0.0002 \\ 0.0307 \end{bmatrix}\tau_N + \begin{bmatrix} 0.0068 \\ 1.3276 \end{bmatrix} \tag{13}$$

2.2. Guidance

As illustrated in Fig. 1, the role of the guidance block in a USV path-following scenario is to determine the desired setpoint course angle value based on the path information and on the current state of the USV.

For the speed dynamics, we did not implement any guidance or control method to simplify the problem. This is still a valid approach because, in the path-following problem, there is no temporal specification (Bibuli et al., 2009) (contrary to trajectory tracking); thus, speed dynamics is less important than steering dynamics. Therefore, for the speed dynamics, we applied a constant target speed V_d to the vehicle's speed controller.

For the steering dynamics, we adopted the vector field guidance (VFG) method proposed by Nelson et al. (2007) to determine the desired course angle. The VFG method was developed to deal with the path-following problem of an unmanned aerial vehicle, but was applied to the maritime domain as well by several researchers (Yiannis and Mitchel; Niu et al., 2016; Woo and Kim, 2016). In the VFG method, the vector field directly indicates the desired direction of the vehicle to move, to follow the given path. On the assumption that the autopilot of the course angle system was first order, Nelson showed a mathematical proof for the Lyapunov stability of the guidance (see Nelson et al., 2007).

Fig. 4 shows an example of the vector field for a linear path. Each of the blue arrows represents the desired course angle of the vehicle when the vehicle is at the designated position. According to the figure,

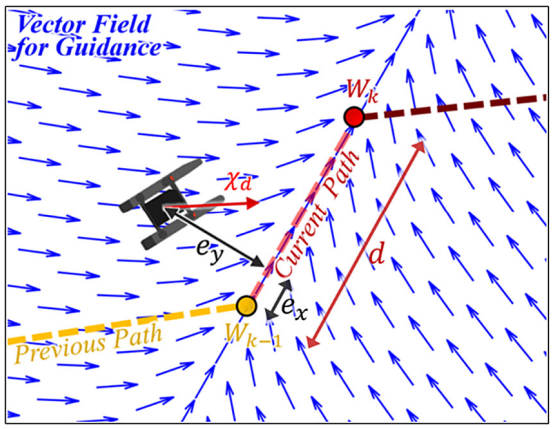

Fig. 5. Path representation using waypoints and the corresponding vector field determined by the VFG model.

the desired course tends to be affected by the path angle χ^{path} and the cross track error e_y (the vehicle's perpendicular distance to the path). When e_y is small, the desired direction is identical to the direction of the path. However, as e_y increases, the difference between the path angle and the desired direction tends to increase as well. Considering this characteristic, Nelson et al. (2007) defined a vector field for a linear path as in (14). According to the equation, the maximum deviation of the course angle from the path angle is limited to χ^{∞}.

$$\chi^d = \chi^{\infty} \cdot tan^{-1}(ke_y) + \chi^{path} \tag{14}$$

A target path for a USV can be represented in various forms, such as a polynomial spline (often used with the Serret–Frenet frame) or a set of waypoints. In this study, we assumed that a path can be considered as a piecewise linear path and can be represented by a set of waypoints. Fig. 5 shows the path representation using a set of waypoints. According to this approach, the target path is represented by using a line of sight (LOS) from the previous waypoint $W_{k-1}(x_{k-1}, y_{k-1})$ toward the current waypoint $W_k(x_k, y_k)$. When d is defined as an Euclidean distance between W_{k-1} and W_k, once the difference between the d and along track error e_x became smaller than certain threshold distance value d_{th}, as $d - e_x < d_{th}$, the target waypoint is changed to the next waypoint. With this approach, a path can always be considered as a linear path and the VFG model for the linear path can be directly applied. To apply the VFG method in path following, we need to identify the vehicle's positional error (the along-track error e_x and the cross track error e_y). The error value can be calculated by using the following process. First, the direction of the path (χ_{path}), the direction from $W_{(k-1)}$ to the USV ($\chi_{W_{k-1}}$) and the distance from the previous waypoint to the USV ($d_{W_{k-1}}$) can be calculated as follows.

$$\begin{aligned} \chi_{path} &= atan\left(\frac{y_k - y_{k-1}}{x_k - x_{k-1}}\right) \\ \chi_{W_{k-1}} &= atan\left(\frac{y_{usv} - y_{k-1}}{x_{usv} - x_{k-1}}\right) \\ d_{W_{k-1}} &= \sqrt{(y_{usv} - y_{k-1})^2 + (x_{usv} - x_{k-1})^2} \end{aligned} \tag{15}$$

Then, using the geometric relationship, we can calculate the along-track error e_x and the cross track error e_y using (16):

$$\begin{aligned} e_x &= cos(\chi_{path} - \chi_{W_{k-1}}) \cdot d_{W_{k-1}} \\ e_y &= sin(\chi_{path} - \chi_{W_{k-1}}) \cdot d_{W_{k-1}} \end{aligned} \tag{16}$$

Once the error variables are identified, the desired course angle can be calculated using (14). The desired course angle is then used as the setpoint value for the steering controller of the USV. In this work, a reinforcement learning-based controller was used for the steering dynamics controller, and a detailed explanation of the controller is provided in Section 3.

3. Reinforcement learning-based controller

Reinforcement learning is one branch of machine learning in which the agent interacts with the environment to find the best policy. In supervised learning, the desired output information corresponds to the input variable, which is provided as a label during the training process. However, in reinforcement learning, the agent directly interacts with the environment without having any information in advance. During the training, the agent performs an action a_t at time t, based on the current state s_t and policy π. As a result of the action a_t, the current state s_t may change according to the transition probability model $p(s_{t+1}|s_t, a_t)$. Based on the evaluation of the state transition, a reward $r(s_t, a_t)$ is received. This process is repeated, and, based on this experience, the policy π is trained. A policy π can be modeled as a stochastic policy $\pi(a|s)$ or a deterministic policy $\pi(s)$. In a reinforcement learning problem, the goal is to find an optimal policy π^* that maximizes the accumulated discounted reward R_t as in (17) (Sutton and Barto, 1998). In the following equation, γ is known as a discount factor, which weighs the future error and has a value between 0 and 1.

$$\begin{aligned} R_t &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdots \\ &= \sum_{k=1}^{\infty} \gamma^k r_{t+k+1} \end{aligned} \tag{17}$$

A policy π can be evaluated by using two value functions. A (state) value function $V^{\pi}(s)$ is defined as the expectation of the accumulated discounted reward R_t while maintaining the policy π. Similarly, an action value function $Q^{\pi}(s_t, a_t)$ is defined as a value function for the specific state and action pair (s_t, a_t).

$$\begin{aligned} V^{\pi}(s_t) &= \mathbb{E}_{\pi}[R_t|s_t] \\ &= \mathbb{E}_{\pi}\left[\sum_{k=1}^{\infty} \gamma^k r_{t+k+1}|s_t\right] \end{aligned} \tag{18}$$

$$\begin{aligned} Q^{\pi}(s_t, a_t) &= \mathbb{E}_{\pi}[R_t|s_t, a_t] \\ &= \mathbb{E}_{\pi}\left[\sum_{k=1}^{\infty} \gamma^k r_{t+k+1}|s_t, a_t\right] \end{aligned} \tag{19}$$

According to the definition of the value functions and the optimal policy π^*, the optimal policy π^* always satisfies the following conditions:

$$\begin{aligned} \pi^* &= \arg\max_{\pi} V^{\pi}(s_t) \\ &= \arg\max_{\pi} Q^{\pi}(s_t, a_t) \end{aligned} \tag{20}$$

To solve the reinforcement learning problem, researchers often use a neural network as an approximator of the value functions. However, the learning process (by updating the temporal difference update) of the reinforcement learning algorithm and the training of the neural network approximator for the value function approximation often interfere with each other, hindering the learning process to be settled. This phenomenon is known as *interference problem* (Carreras et al., 2003) and was considered as the biggest obstacle to the reinforcement learning to be applied in a real-world problem.

Owing to the *interference problem*, training of a large-scale neural network often results in unstable outcomes and, sometimes, even diverges during the learning process. Recently, Mnih et al. (2015) proposed a training method to deal with a large-scale neural network in a reinforcement learning problem. The key factor that Mnih et al. used to eliminate instability was the use of *experience replay* and a *separate target network*. Although the Deep Q Network (DQN) proposed by Mnih et al. (2015) significantly improved the stability and performance of the complex reinforcement learning problem, the method is not suitable for the USV path-following problem because of its discrete action space.

Since the DQN is a type of Q-learning-based reinforcement learning algorithm, it uses a discrete action space. Therefore, the agent can only select the best action from among the limited predefined action

candidates (usually less than 10 action candidates). If the number of action candidates is too small, precise control of the vehicle becomes difficult. Especially at the steady state, the controller tends to periodically change its action (similar to the "bang–bang control"), which is known as chattering. To overcome this limitation, we used a reinforcement learning algorithm that can be applied to a continuous action space. The deep deterministic policy gradient (DDPG) algorithm is an actor-critic structure-based reinforcement algorithm that can deal with a continuous action space. Based on the structure and mathematical foundation of a deterministic policy gradient (DPG) (Silver et al., 2014), the DDPG algorithm can deal with a continuous action space. The method assumes that the target policy π^* is a deterministic policy as μ, but uses a stochastic policy β for exploration. By this assumption, the learning process can be off-policy. In reinforcement learning method dealing with action value function $Q^{\pi}(s_t, a_t)$, updating of the Q-value is performed by using the Bellman equation (22). If there is an assumption that the target policy is deterministic, then the inner expectation can be eliminated as in (22). Since the expectation is dependent on the environment, the policy Q^{μ} could be learned off-policy, which means that the exploration can be separated from the learning process (Silver et al., 2014):

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q^{\pi}(s_{t+1}, a_{t+1})]\right] \tag{21}$$

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, mu(s_{t+1}))\right] \tag{22}$$

When a neural network-based function approximator is parameterized by θ^Q, the approximator can be optimized by minimizing the loss function $L(\theta^Q)$ in (23):

$$L(\theta^Q) = \mathbb{E}_{s_t \sim \rho^{\beta}, a_t \sim \beta, r_t \sim E}[(Q(s_t, a_t|\theta^Q) - y_t)^2], \tag{23}$$

where y_t is known as the temporal difference target and is defined as follows:

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta^Q) \tag{24}$$

By adopting the recently developed techniques in deep reinforcement learning such as *experience replay*, *separate target network*, and *batch normalization*, the DDPG algorithm (Lillicrap et al., 2015) can deal with large-scale neural network approximators.

Fig. 6 shows a schematic diagram of the structure of the DDPG algorithm used in this work. There are two feed-forward neural networks, namely, the actor network and the critic network. Each of the networks is composed of two hidden layers with 400 and 300 node. For the activation of each node, a rectified linear unit function is used. For the updating of the networks, the critic network is updated using the gradient of the loss function $L(\theta^Q)$ in (23), whereas the actor network uses a deterministic policy gradient (Silver et al., 2014), which can be obtained by using (25):

$$\begin{aligned} \nabla_{\theta^{\mu}} J &\approx \mathbb{E}_{s_t \sim \rho^{\beta}}\left[\nabla_{\theta^{\mu}} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^{\mu})}\right] \\ &= \mathbb{E}_{s_t \sim \rho^{\beta}}\left[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s=s_t}\right] \end{aligned} \tag{25}$$

In reinforcement learning, the design of the components of the Markov decision process is one of the most crucial processes. The performance of the algorithm and the speed of the convergence are strongly affected by the correctness of the state space, action space, and reward. In this work, the primary goal of the reinforcement learning-based controller is to minimize the vehicle's cross track error toward the target path(e_y), whereas the desired course angle is provided by the vector field guidance method. According to this, the state space $s \in \mathbb{S}$ is defined as (26)

$$\mathbb{S} = \{\tilde{\chi}, \dot{\tilde{\chi}}, e_y, \dot{e}_y, \delta n_d\}, \tag{26}$$

where $\tilde{\chi}$ is the difference between the course angle of the USV and the desired course angle calculated from the vector field guidance method as $\tilde{\chi} = \tilde{\chi}_d - \tilde{\chi}_{usv}$, e_y is the cross track error, and δn_d is a steering command of the USV. Since $\tilde{\chi}$ only provides angular positional error

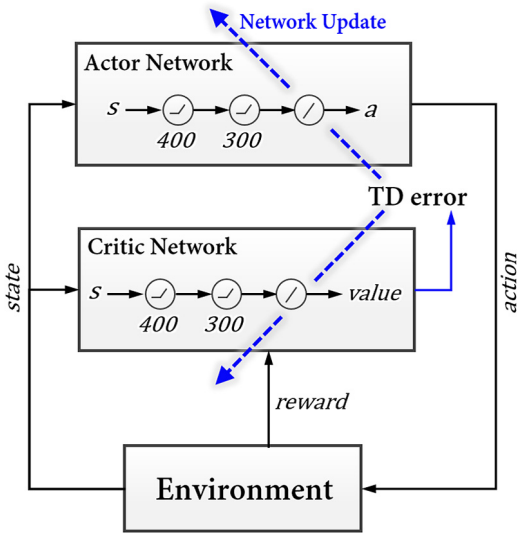

Fig. 6. Schematic diagram of the structure of the control policy networks.

information to the controller, the variable e_y is added to consider the relative positional error information in generating the control input. Moreover, δn_d is included in the state space to provide the current steering command information to prevent the chattering phenomenon. In addition, the time derivative variables $\dot{\tilde{\chi}}$ and $\dot{e}_y$ are included to provide the temporal information to the controller. In order to obtain the state variables during the experiment, we used on-board navigation sensor such as GPS, AHRS to measure the navigational solution. A signal processing filter (Savitzky–Golay filter) is applied to the measurement signal in order to reduce the effect of high-frequency sensor noise. This is an essential process especially for calculating the time derivative variables without magnifying the sensor noise.

Since the reinforcement learning-based controller is designed as a steering controller, the action space $a \in \mathbb{A}$ is defined as (27)

$$\mathbb{A} = \{\delta n_d\}, \tag{27}$$

where δn_d is a steering control command, which defines the RPM commands of the main thrusters as in (9). In this work, since the research scope is limited to the path-following problem, the control action for the reinforcement learning is only focused on the steering control command δn_d. This simplifies the complexity of the reinforcement learning problem and boosts the speed of the convergence.

In the reinforcement learning problem, a reward is used to evaluate the performance of the current policy. Thus, we defined a reward function such that it evaluates the current vehicle status based on the goal of the path-following problem. Since the goal of the path-following problem is to minimize the cross track error and the course angle error without producing chattering, partial reward functions are defined as (28)–(30) and are illustrated in Fig. 7.

$$r_{\tilde{\chi}} = \begin{cases} e^{-k_1 \cdot |\tilde{\chi}|} & \text{if } |\tilde{\chi}| < 90° \\ -e^{-k_1 \cdot (\tilde{\chi} - 180)} & \text{if } \tilde{\chi} \geq 90° \\ -e^{-k_1 \cdot (\tilde{\chi} + 180)} & \text{if } \tilde{\chi} \leq -90° \end{cases} \tag{28}$$

$$r_{e_y} = e^{-k_2 \cdot |e_y|} \tag{29}$$

$$r_{\sigma_{\delta}} = e^{-k_3 \cdot \sigma_{\delta}}, \tag{30}$$

where $\tilde{\chi}$, e_y, and σ_{δ} are the course angle error, cross track error, and standard deviation, respectively, of the recent 20 steering command history values.

As described in Fig. 7, the partial reward function has a positive peak value when each sub-goal is satisfied. For $r_{\tilde{\chi}}$, the partial reward has a maximum value when $\tilde{\chi}$ is equal to 0 and a minimum value when

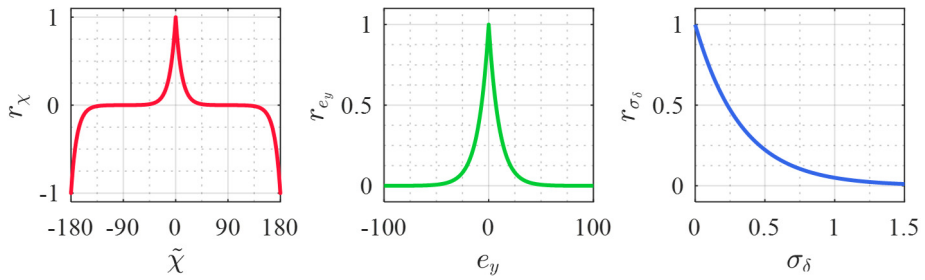

Fig. 7. Partial reward functions designed for USV path following. Each partial reward value reaches the maximum value when $\tilde{\chi}$, e_y, and σ_δ approach 0.

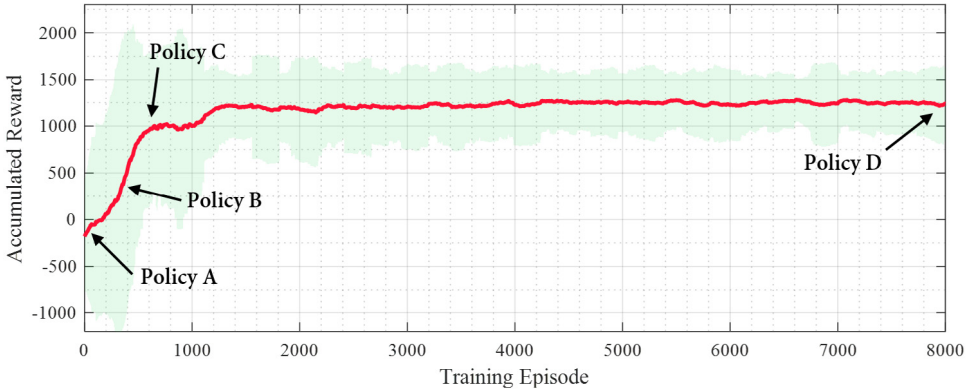

Fig. 8. Moving average (window of 100 episodes) and standard deviation of the accumulated reward value per training episode.

the error is near ±180°. The negative penalty term prevents the vehicle from turning toward the inverse-path direction. For r_{e_y}, the partial reward value reaches a peak value when e_y is equal to 0. Similarly, the value of r_{σ_δ} reaches the maximum value when σ_δ approaches 0. This is the case when the vehicle maintains its steering command statically, without producing any chattering. Overall, the total reward value r is defined as the weighted sum of the partial reward values, as in (31). During the training stage, the weight variables $w_{\tilde{\chi}}$, w_{e_y}, and w_{σ_δ} are selected as 0.4, 0.5, and 0.1 respectively. Combination of the weight variables is determined by using the heuristic approach through repeated trials and errors. In our work, the constant variables in the partial reward function were selected as 0.1, 0.2, and 0.3 for k_1, k_2, and k_3, respectively.

$$r = w_{\chi_e} r_{\chi_e} + w_{e_y} r_{e_y} + w_{\sigma_\delta} r_{\sigma_\delta} \tag{31}$$

In this work, the reinforcement learning-based controller was trained through repeated (up to 8000 episodes and about 750,000 training steps) path-following simulations. During the simulation, the initial USV position, heading angle, and path angle were randomly selected to generate a variety of encounter conditions with the target path. The maximum time step for each episode was limited to 1000 steps, and each episode was finished when the cross track error e_y became larger than 40 m. In the training simulation, the time step was defined as 0.1 s (10 Hz) and the size of the minibatch was set to 64. For the training of the network, the Adam optimizer was used to train both the actor and the critic network. The learning rate was set to 10^{-4} for the actor network and to 10^{-3} for the critic network. The target network transition gain τ was selected as 10^{-3}, and the discount factor γ was selected as 0.99. For the exploration of the training, the Ornstein–Uhlenbeck exploration method discussed in Lillicrap et al. (2015) was used.

Fig. 8 illustrates the learning curve achieved as a result of the training. The red line represents the moving average of the accumulated reward for each episode, and the shaded region represents the standard deviation of the accumulated reward within the moving window. The result showed that the average value of the accumulated reward tended to monotonically increase until it reached about 1300 episodes. After that phase, the average and the standard deviation of the accumulated reward tended to stabilize. According to this learning curve, we can discover the development tendency of the policy as the training proceeds; however, it is still uncertain how the policy works as a controller in a path-following problem. To find out the behavior of the control policies from different training phases, we selected four different training phases and extracted the policy parameter from the actor network. The extracted parameters were then used for the path-following controller, and the performance and behavior of the controller are analyzed in the next section.

Fig. 9. WAM-V platform during the path-following experiment in Pyeongtaek Lake.

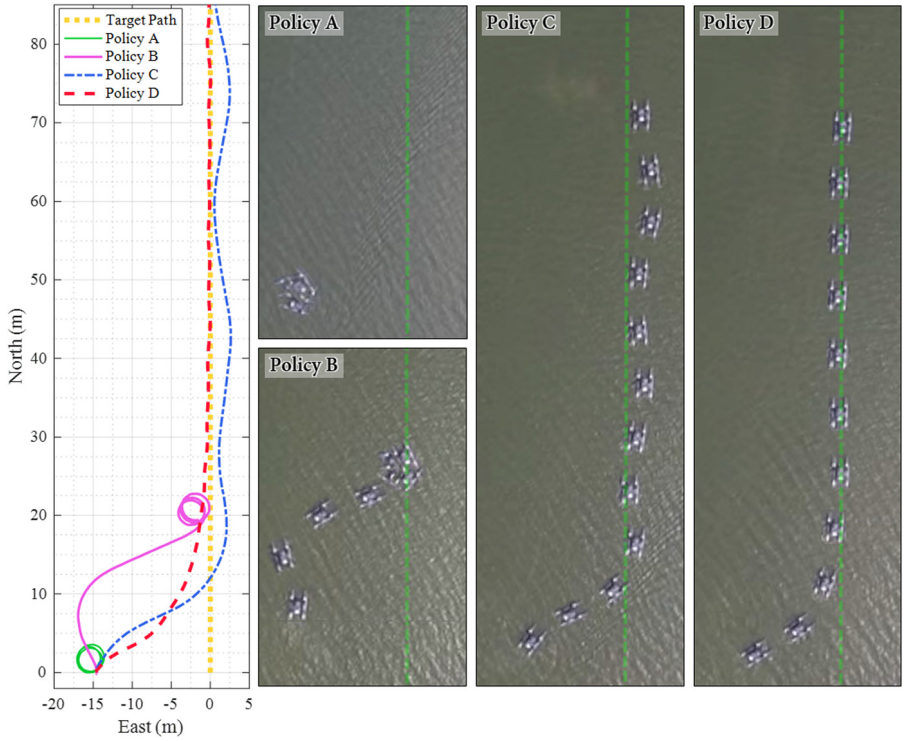

Fig. 10. Trajectory data collected from the linear path-following experiment with various control policies (left) and a video snapshot of the WAM-V platform during the linear path-following experiments (right).

4. Validation

To validate the path-following and self-learning capabilities of the proposed method, we conducted several USV simulations and free-running tests with a full-scale USV. For validation purposes, we selected two different types of path as a target path. First, we selected a linear path as a target path. Since a USV path is often represented by a set of waypoints, a path can be expressed as a piecewise linear path. Thus, the path-following capability of a linear path can be the most fundamental requirement for a USV. Second, a grid search path was selected as a target path. A USV is often used to substitute for a human operator in performing dangerous or tedious repeated operations. Such missions include data collecting mission and object search mission (either an above-water search or an underwater search), which often follow the grid search path. A grid search path may be considered as a piecewise linear path; however, we selected a grid search path since it can cover a dynamic behavior during the path transition of an adjoining piecewise linear path.

To analyze the proposed method's self-learning capability, we extracted a series of control policies (policies A, B, C, and D) during the training process, as illustrated in Fig. 8. The extracted policies were then implemented in the full-scale USV as a steering controller. A number of path-following experiments were conducted with different target path types and various control policies. An experiment was conducted in Pyeongtaek Lake, South Korea, during a calm water condition. For the USV, as mentioned in Section 2, the WAM-V platform was used as a target USV. Note that the control policies were trained by simulation, but were directly implemented in the free-running experiment. We concluded that this was still reasonable since the dynamic model used in the simulation was directly identified from the same USV platform and actuators, using the system identification experiment (Woo et al., 2018). Fig. 9 shows the experimental area and the WAM-V platform during the path-following experiment.

Fig. 10 shows a number of USV trajectories resulting from the linear path-following experiment with different control policies. For each control policy, an identical linear path was used as the target path. In addition, the initial value of the position and heading angle of the vehicle was intentionally controlled, in an attempt to eliminate its effect on the path-following capability. The right side of Fig. 10 shows an aerial snapshot of the USV during the path-following experiment with various control policies. The snapshot pictures were taken sequentially at 4 s of time interval. In this work, we used two error variables to measure the performance of the control policy, namely, the cross track error e_y and the course angle error $\tilde{\chi}$. The cross track error is defined as a lateral positional error that is perpendicular to the path angle, whereas the course angle error is defined as the difference between the USV course angle and the desired course angle calculated from the VFG guidance command. Fig. 11 shows the time history of the cross track error and the course angle error during the path-following experiments, and Fig. 12 shows the corresponding control input during the experiments.

According to the experimental result, as the training proceeded from policy A to policy D, the linear path-following capability tended to develop. At the initial training stage (policy A), the control policy generated a *hardover* maneuver toward the port side, producing a full negative steering command continuously as soon as the path-following mode was switched on. This could cause a repeated turning maneuver of the USV, as indicated by the green line in Fig. 10. After a certain period of training (about 400 more training episodes), control policy B could be achieved. As illustrated in the magenta-colored line in Fig. 10, the control policy tended to track the desired course angle at the initial stage; however, as soon as the course angle error exceed certain threshold value (at approximately 18 s), the vehicle tended to perform a *hardover* toward the port side (see Fig. 12). This indicates that the policy was capable of tracking the desired course angle when it had a negative course angle error; however, once the course angle error reached a positive value, the control policy started to produce a continuous negative steering command, like what policy A did. This phenomenon vanished when the training process reached policy C. According to the USV trajectory in Fig. 10, the USV did not produce a *hardover* steering command anymore. The path-following capability of

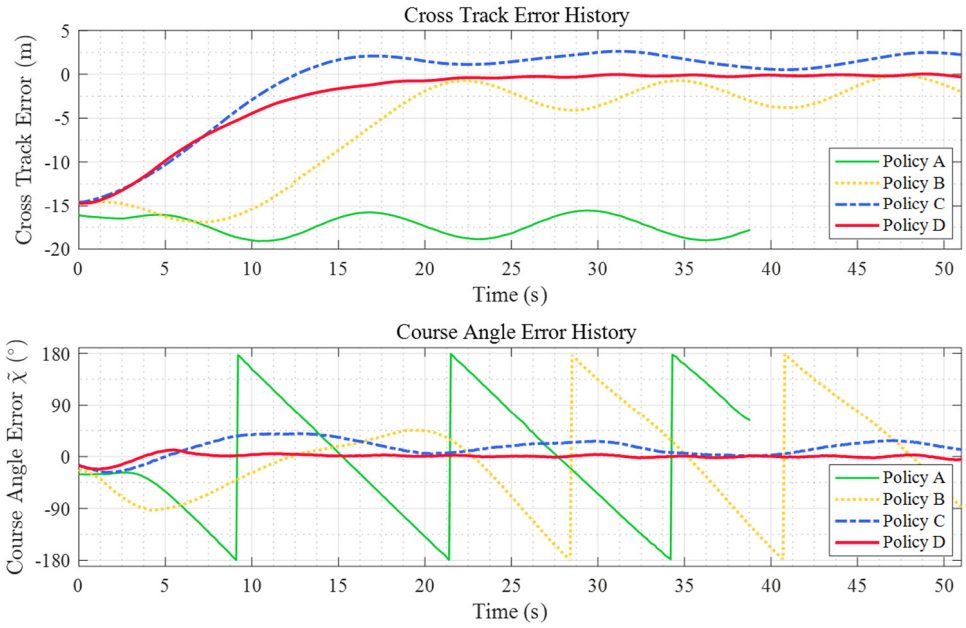

Fig. 11. History of the cross track error e_y and course angle error $\tilde{\chi}$ during the linear path-following experiments with various control policies.

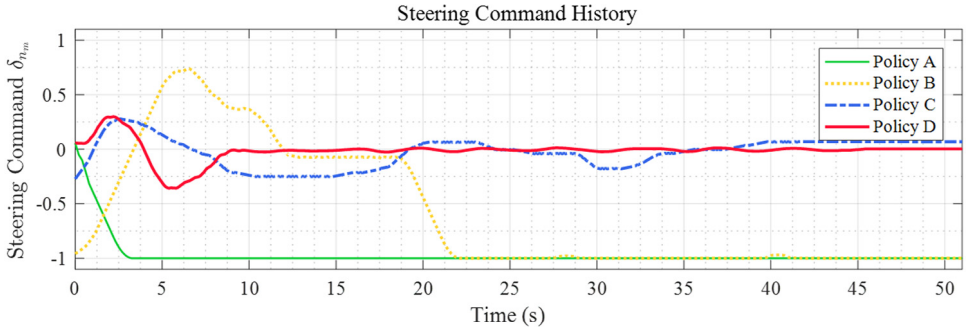

Fig. 12. History of the control input (steering command δn_d) during the linear path-following scenario.

policy C was superior to those of policies A and B; however, this policy tended to have a steady-state error. Once the training stage reached policy D, the path-following capability tended to become robust enough to track a linear path; the control policy did not produce a *hardover* command or a steady-state tracking error. Note that the steering command δ_{n_m} remains nearly zero during the steady state (on dynamic equilibrium point). Such command may be different if there exists side direction force acting on the vehicle such as wind or current loads. Fig. 13 shows the comparison result of the different control policies for the linear path-following problem. Both the cross track and the course angle error tended to decrease as the training proceeded.

So far, we have demonstrated the feasibility of the proposed DDPG-based controller by comparing the path following result of a number of DDPG policy extracted from different learning stages. In order to analyze the performance of the DDPG algorithm, we have conducted linear path following simulations using benchmark controllers as well. For the comparison, we have selected a conventional PID controller and DQN based controller as benchmark controllers. For the control gain of the PID controller, we have heuristically tuned the PID controller until the control performance reaches to a suitable level. Figs. 14–16 represents comparison results of the linear path following simulation using a number of different controllers. As a result of the comparison, the proposed method and DQN based controller tends to have a comparable path following ability as the conventional PID controller. However, the DQN based controller has a chattering tendency which produces constant switching of the control input. Such tendency occurs because

Fig. 13. Result of the performance comparison of the control policies from the linear path-following experiments.

the number of action candidate is limited (e.g. only 10 action candidates), thus in most of the case, a sub-optimal steering command is chosen from the candidates instead of optimal value for that condition.

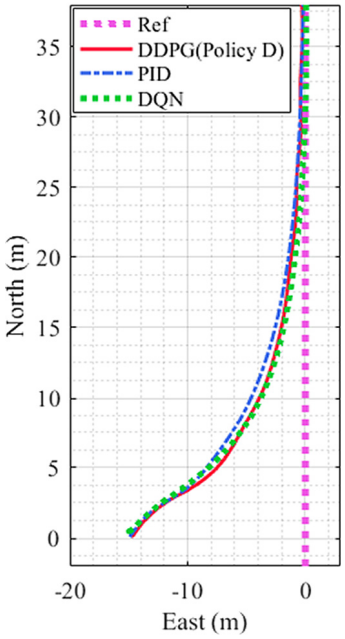

Fig. 14. Comparison result of the trajectory of linear path-following using different types of steering controller.

Table 1
Comparison result of the overall RMS and steady state C.T.E. (Cross Track Error) and C.A.E.(Course Angle Error) during the linear path following validation. Different types and stages of steering controllers are compared.

Performance index	DDPG (A)	DDPG (B)	DDPG (C)	DDPG (D)	DQN	PID
RMS C.T.E. (m)	17.37	7.95	4.35	3.23	3.40	3.61
Steady-State C.T.E. Error (m)	–	–	1.81	0.27	0.48	0.21
RMS C.A.E. (°)	103.28	88.44	20.46	5.10	5.52	4.79
Steady-State C.A.E. (°)	–	–	17.82	0.76	1.76	0.44

Because of this tendency, sometimes the sub-optimal action produces overshoot during the control. To suppress the overshoot, a control input with opposite direction is chosen and by repeating this process, chattering occurs. In the case of the DDPG-based controller, due to the continuous action space characteristics, there exists no such problem, while it follows the target path with comparable control performance as the conventional PID controller. Since the conventional PID controller is designed for SISO (single input–single output) system, the control command of the PID controller depends on only the course angle error. However, in the case of the DRL-based controller, both the cross track error and course angle error are simultaneously used for calculating the control command. Such capability to consider multiple process variable is one of the characteristics of the DRL-based controller. (See Fig. 15.)

Table 1 describes the comparison result of each controller during the linear path following validations.

The dynamic characteristics of the respective control policies during the linear path following could also be found during the search path following. Since the search path was composed of piecewise linear paths, the *hardover* characteristics in policies A and B, and the steady-state error characteristic in policy C were also found in the search path-following experimental data. This is shown in Fig. 17, where Fig. 17 (left) represents the trajectory of the USV during the path-following experiment with various control policies and Fig. 17 (right) shows the simulation result of the USV with the same condition as in Fig. 17 (left) By comparing the figures, we can see that the dynamic behaviors of the USV were similar in both data; in addition, the distinctive characteristics of each control policy could be found in both data. With this phenomenon, we can conclude that the policies trained by the USV simulator can be directly applied to the free-running experiment, although unmodeled dynamics terms or environmental disturbance may exist. In Fig. 18, the cross track error and the course angle error during the experiment are shown, respectively. Like in the linear path following, there was a steady-state error in both the cross track error and the course angle error. One noticeable feature was that the sign of the steady-state errors was always positive (except for the transient phase). This means that the vehicle tended to be on the right side of the current piecewise linear path. Similarly, the direction of the *hardover* turning of policies A and B was always in the same direction (port side turning). This phenomenon could be found in both the linear and the search path case, as well as in both the experiment and the simulation data. In the result of policy D, both the cross track and course angle error tends to be stabilized as soon as the abrupt error is produced by waypoint transition. Overall, we can say that the reinforcement learning-based control policies possessed a control habit, which resulted in a distinctive maneuver under certain conditions. At the initial training stage, such habits resulted in drawbacks in the path-following capability, preventing the vehicle from properly following its target path. However, as the training proceeded, the control habit tended to be eliminated and the control capability tended to develop by itself. This phenomenon could also be found in the learning curve (Fig. 8), where the accumulated reward value increased monotonically as the training proceeded. Although we can be empirically aware of the existence of the control habit and its effect on the path-following performance, because of the massive number of training variables and the complex structure of the policy network, it was impossible to provide an analytical reason as to why such defects were produced and how the control habit generated such distinctive maneuvers. This is one of the drawbacks of the DRL-based controller. In cases involving a massive deep neural network, it is hard to provide an analytical reason as to why such behavior is produced, which makes it more difficult to analyze the produced result. Similarly, there is a limitation on analytical stability analysis of the overall system when using a massive and complex deep neural network. In such case, instead of analytic way, the stability of the controller is often proved by the empirical way such as repeated validation field tests (Rodriguez-Ramos et al., 2019).

According to the learning curve (Fig. 8), the average reward value of policies A, B, and C was not fully developed at the stages. Such underdevelopment could also be found in the activation history of each hidden node in a policy network. Fig. 20 shows the histogram of the probability of each hidden node being activated during the linear path-following scenario. For the fair comparison, we have limited the period of comparison as an initial transient phase (until the course angle error reaches 5% of its initial value). By doing this, the time length of the steady state does not affect to the histogram distribution. In the case of policy A, according to Fig. 20(a), the hidden nodes tended to be either always inactive or always active. This means that only a small number of hidden nodes produced a control action, whereas a number of nodes did not play any role in producing a control action. This tendency tended to gradually vanish as the training proceeded. In the case of policy D, the number of nodes with varying activation probabilities seemed to be uniformly distributed compared to the histogram of policies A, B, and C. This implies that most of the hidden nodes in the policy network participated in generating a control action command. This is a desirable phenomenon because, by doing so, the policy network becomes accessible to the neural network's full capacity of approximating the policy.

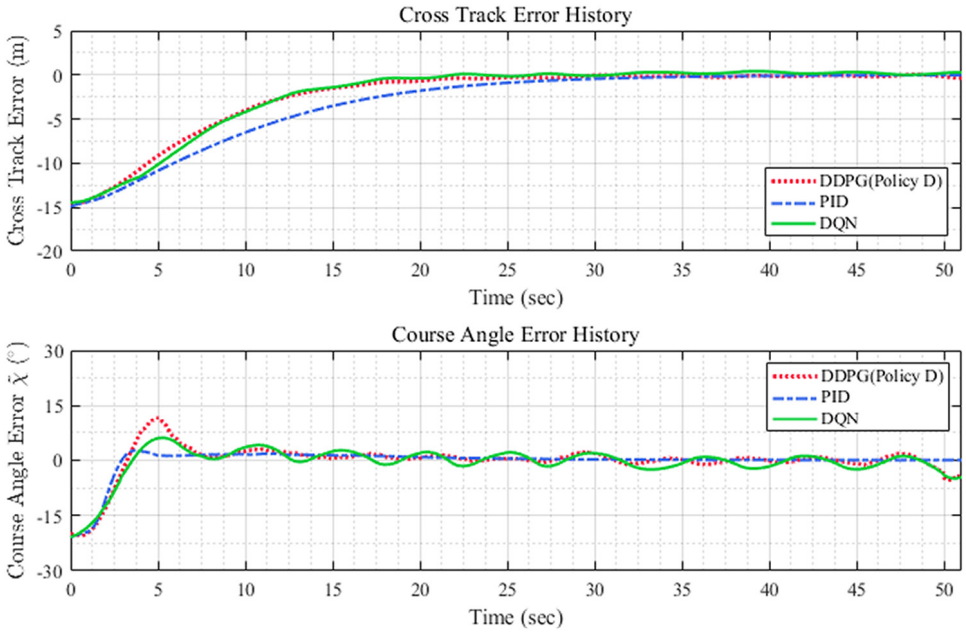

Fig. 15. History of the cross track error e_y and course angle error $\tilde{\chi}$ during the linear path-following using different types of the steering controller.

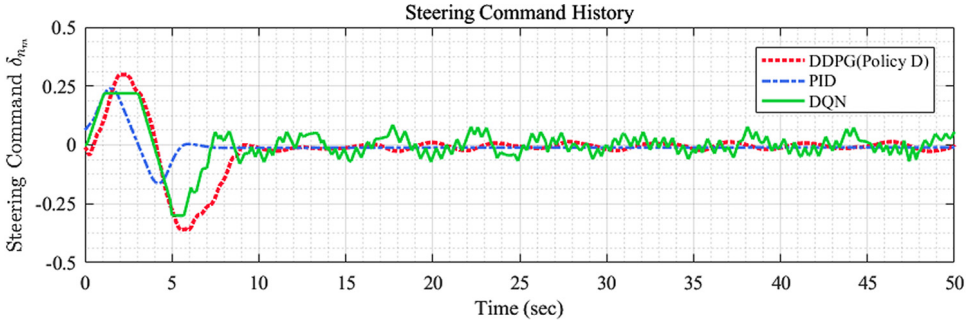

Fig. 16. History of the control input (steering command δn_d) during the linear path-following using different types of steering controller.

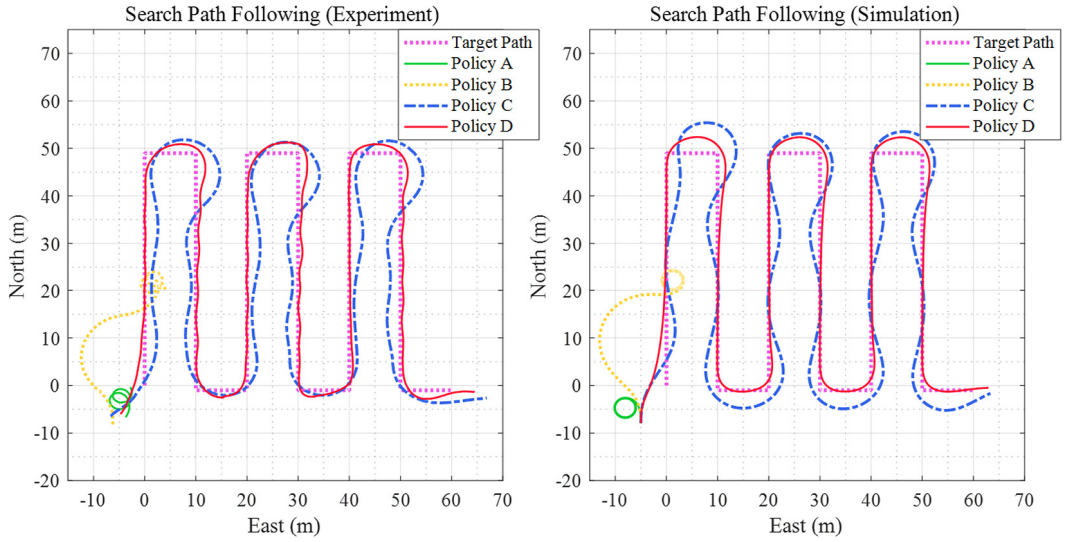

Fig. 17. Trajectory data collected from the grid search path-following experiment (left) and from the corresponding simulation (right) with various control policies.

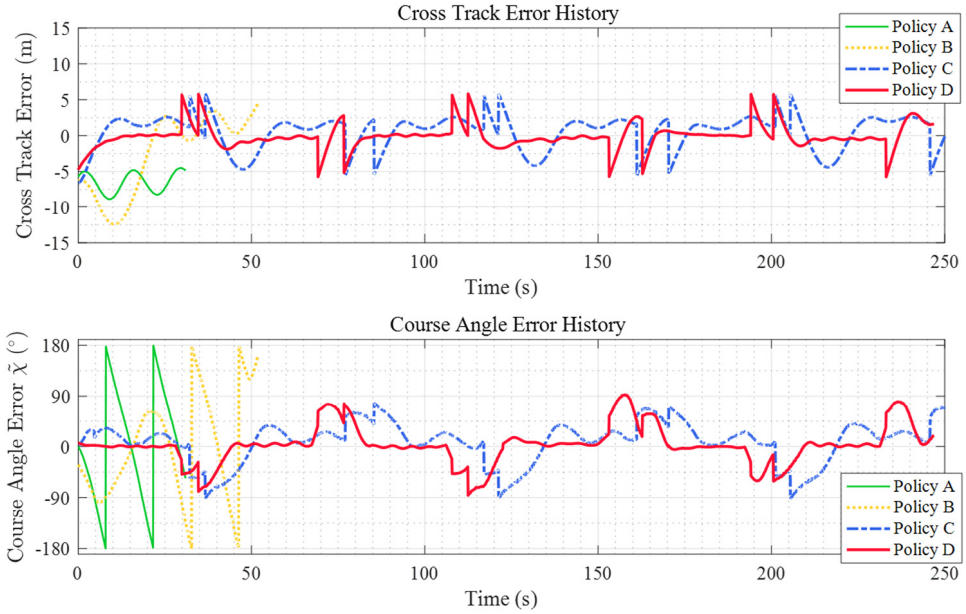

Fig. 18. History of the cross track error e_y and course angle error $\tilde{\chi}$ during the grid search path-following experiments with various control policies.

Fig. 19. History of the control input (steering command δn_d) during the grid search path-following scenario.

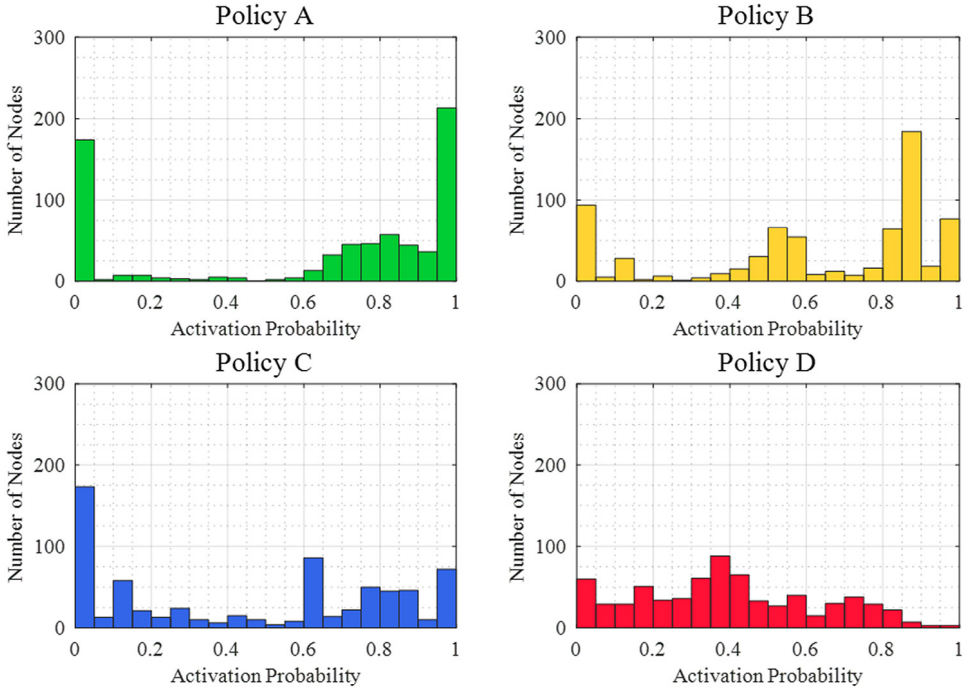

Fig. 20. Histogram of the probability of each hidden node being activated during the linear path-following experiment.

5. Conclusion

In this work, we proposed a deep reinforcement learning-based controller for path following of an unmanned surface vehicle (USV). A deep deterministic policy gradient (DDPG) algorithm was used to construct the steering controller, and with the use of repeated path-following simulation, the learning-based control policy was trained. While the training proceeded, four different control policies were extracted in consecutive order. The extracted control policies were then implemented in a full-scale USV to validate its path-following capability. According to the results of the path-following simulations and experiments, the extracted control policy tended to possess a control habit. At the initial stage of the training, the control habit prevented the vehicle from properly following the target path. This phenomenon was due to the inadequate training of the policy network, and we found that the hidden nodes in the policy network were not uniformly active at this stage of the training. As the training stage continued, the path-following performance tended to develop, resulting in a decrease in the cross track error and the course angle error. The self-learning ability of the USV controller by interacting with the nearby environment is one of the most distinctive features of the proposed method. Such capability can be applied to solve a problem dealing with the uncertain environmental condition.

In future work, it may be beneficial to replace simulation-based training with experimental data-based training or real-time training. In addition, a reinforcement learning-based controller for disturbance rejection could be developed to deal with unmodeled and time-varying environmental disturbance. Moreover, the reinforcement learning-based controller can be extended to much broader research areas such as trajectory tracking or collision avoidance as well.

Acknowledgments

This work was supported by (a) Research Institute of Marine Systems Engineering of Seoul National University, Republic of Korea, (b) Agency for Defense Development (UD160009DD) of Republic of Korea.

References

Bertaska, I.R., 2016. Intelligent Supervisory Switching Control of Unmanned Surface Vehicles (Ph.D. thesis). Florida Atlantic University.

Bertaska, I.R., von Ellenrieder, K.D., 2018. Experimental evaluation of supervisory switching control for unmanned surface vehicles. IEEE J. Ocean. Eng.

Bertram, V., 2008. Unmanned surface vehicles-a survey. Skibsteknisk Selskab, Copenhagen, Denmark 1, 1–14.

Bibuli, M., Bruzzone, G., Caccia, M., Lapierre, L., 2009. Path-following algorithms and experiments for an unmanned surface vehicle. J. Field Robotics 26 (8), 669–688.

Carreras, M., Ridao, P., El-Fakdi, A., 2003. Semi-online neural-Q/spl I. bar/leaming for real-time robot learning. In: Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, vol. 1. IEEE, pp. 662–667.

Cheng, Y., Zhang, W., 2018. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. Neurocomputing 272, 63–73.

Corfield, S., Young, J., 2006. Unmanned surface vehicles-game changing technology for naval operations. IEE Control Eng. Ser. 69, 311.

Cui, R., Yang, C., Li, Y., Sharma, S., 2017. Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning. IEEE Trans. Syst. Man Cybern. Syst. 47 (6), 1019–1029.

De Paula, M., Acosta, G.G., 2015. Trajectory tracking algorithm for autonomous vehicles using adaptive reinforcement learning. In: OCEANS'15 MTS/IEEE Washington. IEEE, pp. 1–8.

Fossen, T.I., 2002. Marine control system-guidance, navigation and control of ships, rigs and underwater vehicles. Mar. Cybemetics.

Garus, J., Zak, B., 2010. Using of soft computing techniques to control of underwater robot. In: Methods and Models in Automation and Robotics (MMAR), 2010 15th International Conference on. IEEE, pp. 415–419.

Kucik, D., Reconnaissance using unmanned surface vehicles and unmanned micro-aerial vehicles, Google Patents, US Patent 6,712,312, 2004.

Lekkas, A.M., Fossen, T.I., 2012. A time-varying lookahead distance guidance law for path following. Arenzano, Italy.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Lu, Y., Zhang, G., Sun, Z., Zhang, W., 2018. Robust adaptive formation control of underactuated autonomous surface vessels based on MLP and DOB. Nonlinear Dynam. 94 (1), 503–519.

Magalhães, J., Damas, B., Lobo, V., 2018. Reinforcement learning: The application to autonomous biomimetic underwater vehicles control. In: IOP Conference Series: Earth and Environmental Science, vol. 172. (1), IOP Publishing, pp. 12–19.

Majohr, J., Buch, T., 2006. Modelling, simulation and control of an autonomous surface marine vehicle for surveying applications Measuring Dolphin MESSIN. IEE Control Eng. Ser. 69, 329.

Meng, W., Guo, C., Liu, Y., Yang, Y., Lei, Z., 2012. Global sliding mode based adaptive neural network path following control for underactuated surface vessels with uncertain dynamics. In: Intelligent Control and Information Processing (ICICIP), 2012 Third International Conference on. IEEE, pp. 40–45.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529.

Mu, D., Wang, G., Fan, Y., Sun, X., Qiu, B., 2017. Adaptive LOS path following for a podded propulsion unmanned surface vehicle with uncertainty of model and actuator saturation. Appl. Sci. 7 (12), 1232.

Naeem, W., Sutton, R., Chudley, J., Modelling and control of an unmanned surface vehicle for environmental monitoring, in: UKACC International Control Conference, August, Glasgow, Scotland, 2006.

Nelson, D.R., Barber, D.B., McLain, T.W., Beard, R.W., 2007. Vector field path following for miniature air vehicles. IEEE Trans. Robot. 23 (3), 519–529.

Niu, H., Lu, Y., Savvaris, A., Tsourdos, A., 2016. Efficient path following algorithm for unmanned surface vehicle. In: OCEANS 2016-Shanghai. IEEE, pp. 1–7.

Rodriguez-Ramos, A., Sampedro, C., Bavle, H., De La Puente, P., Campoy, P., 2019. A deep reinforcement learning strategy for UAV autonomous landing on a moving platform. J. Intell. Robot. Syst. 93 (1–2), 351–366.

Shin, J., Kwak, D.J., Lee, Y.-i., 2017. Adaptive path-following control for an unmanned surface vessel using an identified dynamic model. IEEE/ASME Trans. Mechatronics 22 (3), 1143–1153.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M., Deterministic policy gradient algorithms, in: ICML, 2014.

Sonnenburg, C.R., Woolsey, C.A., 2013. Modeling, identification, and control of an unmanned surface vehicle. J. Field Robotics 30 (3), 371–398.

Sutton, R.S., Barto, A.G., 1998. Introduction to Reinforcement Learning, vol. 135, MIT Press Cambridge.

Woo, J., 2018. Collision Avoidance for an Unmanned Surface Vehicle Using Deep Reinforcement Learning (Ph.D. thesis). Seoul National University.

Woo, J., Kim, N., 2016. Vector field based guidance method for docking of an unmanned surface vehicle. In: The Twelfth ISOPE Pacific/Asia Offshore Mechanics Symposium. International Society of Offshore and Polar Engineers.

Woo, J., Park, J., Yu, C., Kim, N., 2018. Dynamic model identification of unmanned surface vehicles using deep learning network. Appl. Ocean Res. 78, 123–133.

Yiannis, P., Mitchel, W., Operations architecture and vector field guidance for the river scout subscale unmanned surface vehicle, in: Defence and Homeland Security Symposium, 2013, pp. 55–60.

Yoo, B., Kim, J., 2016. Path optimization for marine vehicles in ocean currents using reinforcement learning. J. Mar. Sci. Technol. 21 (2), 334–343.

Zhang, G., Deng, Y., Zhang, W., Huang, C., 2018. Novel DVS guidance and path-following control for underactuated ships in presence of multiple static and moving obstacles. Ocean Eng. 170, 100–110.

Zhang, R., Tang, P., Su, Y., Li, X., Yang, G., Shi, C., 2014. An adaptive obstacle avoidance algorithm for unmanned surface vehicle in complicated marine environments. IEEE/CAA J. Autom. Sin. 1 (4), 385–396.

Zhu, J., Wang, J., Zheng, T., Wu, G., 2016. Straight path following of unmanned surface vehicle under flow disturbance. In: OCEANS 2016-Shanghai. IEEE, pp. 1–7.