End-to-end Trajectory Tracking Algorithm for Unmanned Surface Vehicle Using Reinforcement Learning

Hongdong Wang
Shanghai Jiao Tong University
23 PUBLICATIONS 31 CITATIONS
SEE PROFILE

Hong Yi
Shanghai Jiao Tong University
363 PUBLICATIONS 8,004 CITATIONS
SEE PROFILE

Proceedings of the Twenty-ninth (2019) International Ocean and Polar Engineering Conference
Honolulu, Hawaii, USA, June 16-21, 2019
Copyright © 2019 by the International Society of Offshore and Polar Engineers (ISOPE)
ISBN 978-1 880653 85-2; ISSN 1098-6189
www.isope.org

End-to-end Trajectory Tracking Algorithm for Unmanned Surface Vehicle Using Reinforcement Learning

Kefan Jin, Hongdong Wang, Hong Yi
[1] MOE Key Laboratory of Marine Intelligent Equipment and System, Shanghai Jiao Tong University Shanghai, China
[2]State Key Laboratory of Ocean Engineering, Shanghai Jiao Tong University Shanghai, China
Shanghai, China

ABSTRACT

Autonomous motion control of USVs, especially in complex marine conditions, is always a fundamental problem. Conventional methods consider the USV hydrodynamic model and the influences of environmental disturbance separately. However, due to the randomness of wind, wave and current, the accumulative error of each model can be large. To address this issue, this paper presents an end-to-end USV tracking control method via deep reinforcement learning, where a modern Reinforcement learning algorithm Actor-Critic is adopted. Given no prior knowledge of the dynamical system, the proposed method takes as input the information of environment (e.g., speed of wind and flow, etc.), ship and target trajectory, then produces the ship control signal (i.e., rudder angle and forward momentum) directly. We further propose a customized reward function to appraise the performance of ship agent. The presented simulation results demonstrate that this novel algorithm performs well in tracking tasks under complex marine conditions which is designed to change constantly.

KEY WORDS: unmanned surface vehicle, reinforcement learning, deep learning, trajectory tracking

INTRODUCTION

In recent years, unmanned surface vehicle (USV) is playing an increasingly import role in civil and military field. With the advantage of low-cost and multifunction, USV can easily be applied to many scenarios, like oceanographic measurement, ocean resource exploration, marine cruise and so on. In all kinds of tasks, trajectory tracking ability is essential for USVs and influences their performance. Traditionally, people build the dynamical model to calculate the hydrodynamic force (Fossen, Breivik, & Skjetne, 2003). Ivar-André F.Ihle applied passivity-based method to this problem(Ihle, Arcak, & Fossen, 2007). Marco Bibuli use Lyapunov-based guidance to guarantee the convergence of path-following(Bibuli, Bruzzone, Caccia, & Lapierre, 2009). And many other control methods are developed(Ashrafiuon, Muske, & McNinch, 2010; Wang, Gao, Sun, & Zheng, 2018; Xiang, Yu, Lapierre, Zhang, & Zhang, 2018). When working in real world scenarios, the effect of the environment (i.e., wind, flow and wave) can not be neglected. A commonly used method is to build model for each force and much effort has been done (Fossen, 2012; Fossen & Lekkas, 2017; Fossen, Pettersen, & Galeazzi, 2015). However, it is notoriously challenging to calculate the effect of these disturbance precisely and there are always errors with the model, especially when it comes to wave, which is so complex that its real dynamics model can not be totally built yet. Moreover, the randomness of the environment will also increase the model errors. And when synthesizing all the models, the total error can even be larger, which may leads to the poor performance of USV in realistic environment.

On the other side, the artificial intelligence technique has made great progress in recent years. Mnih, Volodymyr and Koray Kavukcuoglu design a deep Q-network agent to play Atari 2600 games, whose performance is able to compete that of a professional human games tester(Mnih et al., 2013; Mnih et al., 2015). AlphaGo(Silver et al., 2016), a computer Go agent who defeat the best human player, can be regarded as a landmark, while its improved version AlphaGo Zero and AlphaZero (Silver, Hubert, et al., 2017; Silver, Schrittwieser, et al., 2017) can even be more capable. And the success of Libratus(Brown & Sandholm, 2017), the first AI to defeat top humans in heads-up no-limit Texas hold'em poker, also suggest that artificial intelligence agent is able to deal with imperfect-information tasks. All the progress above are based on deep reinforcement learning (DRL) technique. Reinforcement learning is about an agent interacting with the environment, learning an optimal policy, by trail and error, for sequential decision making problems in a wide range of fields in both natural and social sciences, and engineering(Sutton & Barto, 2018). Andrew Ng first successfully use a reinforcement learning algorithm to realize aerobatic helicopter flight(Abbeel, Coates, Quigley, & Ng, 2007). Jie Tan and Tingnan Zhang applied the RL-based actuator model to learning agile locomotion for quadruped robots(Tan et al., 2018). Pete Florence managed to train the robot arm quickly for a wide variety of previously unseen and potentially non-rigid objects(Florence, Manuelli, & Tedrake, 2018).

In this paper, we present an end-to-end USV motion control method via deep reinforcement learning where a modern reinforcement learning algorithm A3C(Mnih et al., 2016) is adopted. Instead of building the dynamical models for each part of drift forces, the proposed method construct multiple agents. These agents will be trained asynchronously on multiple instances of the environment in parallel and every agent consist of two parts named Actor and Critic where Actor is designed to realize the task of locomotion for USV and Critic to evaluate its condition. In this way, given specific ship form parameters the control signal can be produced directly with the change of wind, wave, current and target deviation as input, thus avoiding the cumulative errors derived from every model.

Given the time difference between when a control signal is sent and when it take effect completely, the n-step learning algorithm is adopted. We further propose a customized reward function to evaluate its performance. Finally, this algorithm is demonstrated by simulations. The results show that this end-to-end trajectory tracking algorithm can work in a variety of environments and has strong robustness.

SYSTEM DESCRIPTION

In this paper, we assume that the marine craft experiences motion in 3 degrees of freedom (DOFs), and use horizontal plane model (surge, sway and yaw) to describe the system. Hence, the items $g(\eta)$ and g_0 are ignored. The velocity vector of the USV is $\nu =$ [u, v, r], and the position vector is $\eta =$ [x, y, ψ]. We denotes the velocity in surge and sway as u and v, the angular velocity of yaw as r in body coordinates, the position coordinate of USV as [x, y], and heading of USV as ψ in the earth-fixed inertial frame. （Fig. 1）

Dynamic Model of USV

To facilitate designed control, the dynamic model is decoupled, and the simulator of USV is established. The marine craft equations of motion(Fossen, 2011) can be written as:

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

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) + g_0 = \tau + \tau_{wind} + \tau_{wave} + \tau_{flow} \tag{2}$$

where $J(\eta)$ is the rotation matrix corresponding to the z axes, M is the rigid-body inertia matrix, $C(\nu) = C_{RB}(\nu) + C_A(\nu_r)$ is the rigid-body Coriolis and centripetal forces matrix (including added mass), $D(\nu)$ is the damping matrix, $g(\eta)$ is a vector of generalized gravitational and buoyancy forces , g_0 is the collected static restoring forces and moments due to ballast systems and water tanks, τ is the generalized forces, τ_{wind} is the environmental wind forces, τ_{wave} is the environmental wave forces, τ_{flow} is the environmental flow forces.

By ignoring the heave, roll and pitch, the rotation matrix can be expressed as:

$$J(\eta) = \begin{bmatrix} \cos\psi & \text{-}\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

And model (2) is expressed as:

$$M\dot{\nu} + C_{RB}(\nu)\nu + N(\nu_r)\nu_r = \tau + \tau_{wind} + \tau_{wave} + \tau_{flow}$$

where added mass Coriolis and centripetal terms together with hydrodynamic damping terms are collected into matrix $N(\nu_r)$, for they are both hydrodynamic forces.

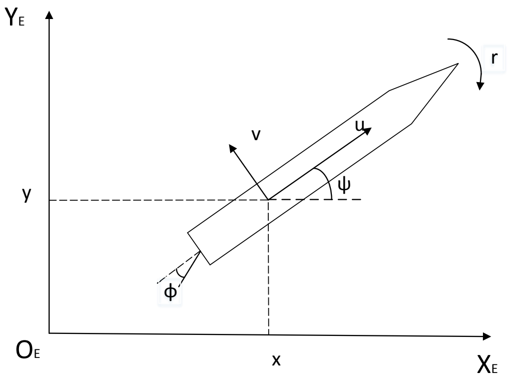


Fig. 1 Diagram of angles of USV

$$M = \begin{bmatrix} m - X\dot{u} & 0 & 0 \\ 0 & m - Y\dot{v} & mx_g - Y\dot{r} \\ 0 & mx_g - Y\dot{r} & I_z - N\dot{r} \end{bmatrix} \tag{4}$$

$$C_{RB}(\nu) = \begin{bmatrix} 0 & 0 & -m(x_g r + v) \\ 0 & 0 & mu \\ m(x_g r + v) & -mu & 0 \end{bmatrix} \tag{5}$$

The expression for $N(\nu_r)$ depend on how the dissipative forces are modeled:

$$N(\nu_r)\nu_r = C_A(\nu_r)\nu_r + D(\nu_r)\nu_r \tag{6}$$

where

$$C_A(\nu) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}}v + Y_{\dot{r}}r \\ 0 & 0 & -X_{\dot{u}}u \\ -Y_{\dot{v}}v & X_{\dot{u}}u & 0 \end{bmatrix} \tag{7}$$

$$D(\nu) = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix} \tag{8}$$

Problem Formulation

The main objective of this paper is to realize the trajectory tracking task of unmanned surface vehicle, that is, the ship have to reach the destination by prescribed route with external force disturbance consists of wind, flow and wave. In this task, we divide the curvilinear target course into several straight course. For each part of the course, the starting point coordinates and end point coordinates are available. Then, we adopt a modern reinforcement learning algorithm Asynchronous Advantage Actor-Critic (A3C), where the agent is designed to make sequences of control strategies, and a customized reward function is designed to train the agent. Let us assume that the states of disturbance, including wind speed, wind direction, flow rate, flow direction, significant wave height, wave length and wave encounter angle, are acquirable, which are denoted by v_w , φ_w , v_f , φ_f , ζ , λ and χ respectively. We also need the ship velocity vector, velocity change, ship

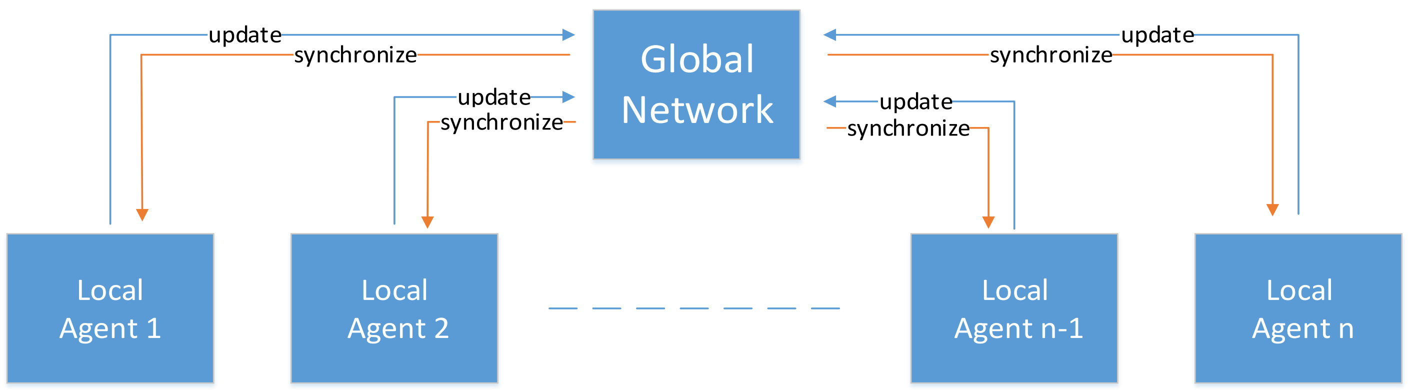


Fig. 2 Asynchronous Advantage Actor-Critic description

position vector, course deviation, path deviation and rudder angle, which are denoted by $\nu =$ [u, v, r], $\Delta\nu =$ [Δu, Δv, Δr], $\eta =$ [x, y, ψ], $\Delta\psi$, d and φ.

To learn the navigation in such complex environment, we use an end-to-end trajectory tracking algorithm based on reinforcement learning. One advantage of this approach is that control signals are not calculated by several models of different kinds of disturbance, but learnt together, thus avoiding the accumulated error from each model. And this algorithm can be easily applied to other ships by changing the ship form parameters. But we also have some problems. First, because there is only one goal location, the rewards are often sparse. Second, compared to cars and drones, the control signal can not take effect on the trajectory. It is a slow process, which require the agent to have good ability of prediction.

To improve the training efficiency, we augment the reward with auxiliary task that provide denser training signals to increase training efficiency. We further propose a customized reward function. We evaluate our approach with random sea condition, and compare the results of the performance, which worked with and without changing states of disturbance respectively.

TRAJECTORY TRACKING ALGORITHM

Reinforcement learning (RL) is about an agent interacting with the environment, learning an optimal policy, by trail and error, for sequential decision making problems in a wide range of fields in both natural and social sciences, and engineering (Sutton & Barto, 2018). The RL agent interacts with environment continuously. At each time step t, given the observation s_t, the agent take an action a_t, which will lead to the next observation s_{t+1}. Then, according to the action and observation pairs, the environment will return a scalar reward r_t, which is the evaluation of behavior and used for training. And we take the accumulated return from time t as $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, where $\gamma \in (0,1)$ denotes the discount factor. After being trained, the agent is expected to learn to choose the action a to maximize the total accumulated return R_t.

A3C Algorithm

The end-to-end trajectory tracking problem is addressed with the Asynchronous Advantage Actor-Critic (A3C) algorithm (Mnih et al., 2016). Traditional Deep RL algorithms are usually based on experience replay, which need more memory and computation per real interaction, and as an off-policy learning algorithm, it updates from data generated by an older policy. In the A3C algorithm, we asynchronously execute multiple agents in parallel on multiple instances of the environment （Fig. 2）, with only one global network synthesizing the data from each agents, which is also presented as the ultimate outcome. During the training process, all the agents explore its own environment respectively. At time step t, agents will update the parameters of the global network in turn, and global network also synchronize its parameters with all the agents in parallel.

Each agent relies on both a policy function $\pi(a_t \mid s_t;\theta)$ and a value function $V(s_t;\theta')$.（Fig. 3） Given the observation state s_t, the action is drawn from the policy function distribution: $a_t \sim \pi(\cdot \mid s_t;\theta)$, with policy network parameters θ, referred to as Actor. Then this action will lead to a new state s_{t+1} and a reward r from environment. As we know, this transition (s_t, a_t, s_{t+1}) must be subject to an unknown state transition function, and it will receive a reward from environment, which we expected to be approximated by the value function $V(s_t;\theta')$, with value network parameters θ', referred to as Critic. In a training process, the transition (s_t, a_t, r, s_{t+1}), which is resulted from Actor will be the training data of Critic. Meanwhile, the updating of Actor's parameters is based on the scalar evaluation output from Critic.

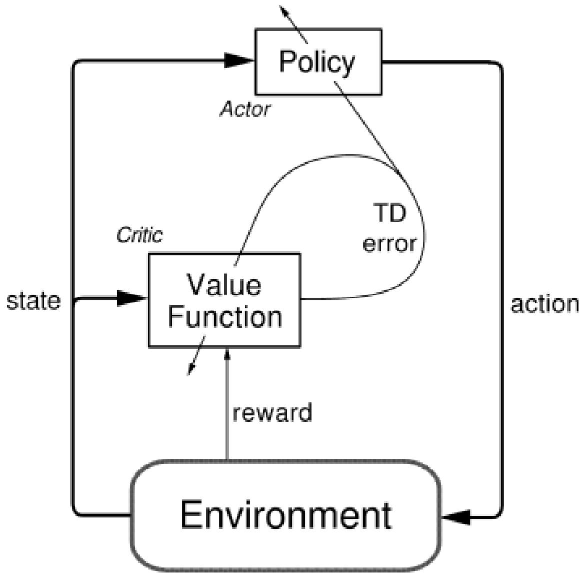


Fig. 3 Actor-Critic description

Reward Function

In the trajectory tracking task, reward function is a key part of the algorithm designing. A basic reward will be given once the ship reach the destination. And we also calculate the shortest straight line distance between the ship's current horizontal position $[x, y]$ and target line as path deviation d. Higher the deviation, lower the reward. In this task, shortening the ship deviation from target line is of the highest priority.

Theoretically, this reward function is available for training process, but it is far from high efficiency. In order to improve the training efficiency, we further propose two auxiliary rewards. First, we consider the course angle as an auxiliary task. Here the line-of-sight (LOS) guidance system is adopted. Let the ship's current horizontal position $p=[x, y]$ be the center of a max deviation circle with radius of n ship lengths (nL_{pp}). The former intersection p_{los} of the circle and target course line is set to be the temporary target, and the vector from p to p_{los} is the temporary target course with the target course angle is denoted by $\hat{\psi}$. (Fig. 4) A reward will be given if the deviation of ship's current course ψ and target course $\hat{\psi}$ is kept at a small value: $R_{course} = \cos(\psi - \hat{\psi})$

However, under some conditions, this reward function may lead to an extremely low speed to gain higher reward, which is not the desired behavior. To address this issue, the shortened distance to destination at each time step t is proportional to the reward to encourage the agent to arrive at the goal as soon as possible. We denote the distance to destination at time step t as l_t.

$$R_{close} = l_{t-1} - l_t \tag{9}$$

We can now write the reward function as:

$$\begin{aligned} R &= A + \lambda_1 |d| - \lambda_2 R_{close} - \lambda_3 R_{course} \\ &= A + \lambda_1 |d| - \lambda_2 \cos(\psi - \hat{\psi}) - \lambda_3 (l_{t-1} - l_t) \end{aligned} \tag{10}$$

where $A > 0, \lambda_1 > 0, \lambda_2 > 0, \lambda_3 > 0$ are the parameters to be tuned, and the d denotes the deviation from the target path.

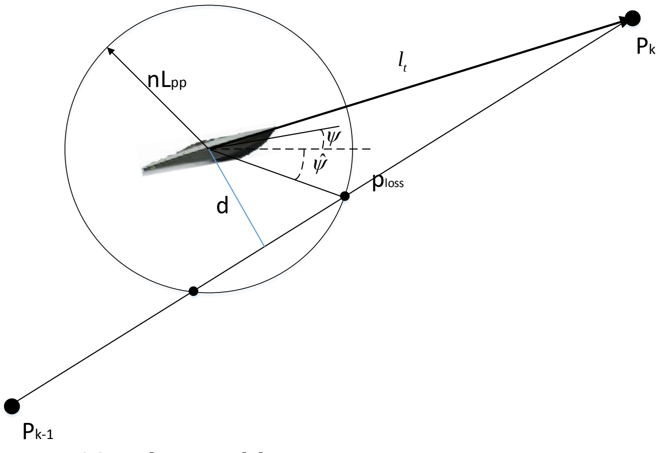


Fig. 4 LOS guidance model

Algorithm Designing

The end-to-end trajectory tracking control method is addressed with the Asynchronous Advantage Actor-Critic (A3C) algorithm (Mnih et al., 2016). First, we construct multiple asynchronous agents. Unlike the Gorila framework(Nair et al., 2015), we do not need multiple machines. Instead, we use only one machine with multiple CPU threads. All the agents are distributed among different threads respectively. This method saves time for gradients and parameters transfer between machines, improving the operating speed. And it also lower the hardware cost, making it available for most people. In this task, we construct 8 agents to be trained in parallel. During the training process, each agent will explore its own environment, with different wind, wave, flow conditions and different target course. And the parameters updating process of global network are operated by multiple agents, reducing the overall change correlation in time. In this way, we do not need the huge replay buffer, which requires huge memory and more computation per real interaction.

Every agent consists of Actor network $\pi(a_t | s_t; \theta)$ and Critic network $V(s_t; \theta')$, where the wind speed, wind direction, flow rate, flow direction, significant wave height, wave length, wave encounter angle, ship velocity vector, velocity change, ship position vector, course deviation, path deviation and rudder angle are taken as state. After receiving a observation state s_t, Actor produces the ship control signal a (e.g., rudder angle and ship thrust). Then the scalar valuation is given by Critic. In this algorithm, the functions of Actor and Critic are similar in a way, thus the two network share the first few layers of the networks, and these two network will also be optimized jointly. We also clip the origin reward to enhance the system robustness.

```
Algorithm 1
Randomly initialize global critic network with parameter $\theta'$
Randomly initialize global actor network with parameter $\theta$
Initialize total step counter T ← 0
Initialize multiple agents networks
Initialize update step n
repeat
  T ← T+1
for each agents
    Initialize local temporal replay buffer $R$
    Initialize local step counter t ← 0
    Reset environment
    Receive initial observation state $s_1$
    repeat
      Take action $a$ with current policy $\pi$
      Receive new state $s_{t+1}$ and reward $r$
      Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
      if t mod n ==0 or $s_{t+1}$ is terminal state then
        $y = \begin{cases} 0 & \text{for terminal state} \\ V(s_t) & \text{for non-terminal state} \end{cases}$
        for $i \in \{t-1, \ldots, t_{start}\}$ do
          $y \leftarrow \gamma y + r_i$
          Update global critic: $\theta' \leftarrow \theta' - \alpha \frac{\partial (y - V(s_i))^2}{\partial \theta}$
          Update global critic :
            $\theta \leftarrow \theta + \alpha (y - V(s_i)) \nabla_\theta \log \pi(a_t | s_t) + \beta \nabla_\theta H(\pi(\cdot | s_t))$
        end for
        Clear $R$
        Synchronize local parameter with global parameter
      $t \leftarrow t+1$
      end if
    until $s_{t+1}$ is terminal state
until $T > T_{\max}$
```

Fig. 5 Algorithm description

Unlike other tasks, the motion control of ship has a certain delay, the ship can not change its trajectory immediately once the control signals are given. After the control command is issued, it takes some time to take effect. However, the traditional method, one-step learning, updates the parameters θ and θ' with only one reward r_t being considered:

$$\theta' \leftarrow \theta' + \alpha \frac{\partial (r+\gamma V(s_{t+1}) - V(s_t))^2}{\partial \theta} \tag{11}$$

$$\theta \leftarrow \theta + \alpha (r+\gamma V(s_{t+1}) - V(s_t)) \nabla_\theta \log \pi(a_t \mid s_t) \tag{12}$$

This will lead to a result that the effect of the control can not match the immediate reward, and a control signal will also have a lasting impact to the ship movement. To solve this problem, we adopt the n-step learning method, which take account of the continuing influence of one signal. Not only immediate reward, the n rewards during n steps after the action being taken are considered:

$$\theta' \leftarrow \theta' - \alpha \frac{\partial (y - V(s_i))^2}{\partial \theta} \tag{13}$$

We also define the total moving reward RT_x (Fig. 6), which reflect the intelligence of the agents at xth epoch as:

$$\theta \leftarrow \theta + \alpha (y - V(s_i)) \nabla_\theta \log \pi(a_t \mid s_t) + \beta \nabla_\theta H(\pi(\cdot \mid s_t)) \tag{14}$$

where

$$y = \Sigma_{T=t}^{t+n-1} \gamma^{T-t} r_T + \gamma^T V(s_{t+n}) \tag{15}$$

y denotes the cumulated sum during future n steps, with discounted factor $0<\gamma<1$, α is the learning rate. $H(\cdot)$ is an entropy regularizer, which is set to encourage the exploration of agents, and β is the regularizer factor.

EXPERIMENT RESULTS

In this section, some examples are given to illustrate the effectiveness of the algorithm. We use Tensorflow deep learning framework to construct the system, and the dynamics of the ship are unknown to the agents. The parameters are set to be same with(Moreira, Fossen, & Soares, 2007). The radius of the max deviation circle in LOS guidance system is set to be 1m, and once the ship is so far from the target course that the target course line dose not intersect the circle, this training will end with a negative reward. So, each training epoch will end if the course deviation continue to be larger than 20m for 20 seconds or the ship reach the destination with deviation maintaining small. In this experiment, we set the target course to be straight. After every training epoch, the environment will be reset, that means the ship will be given new random start point and target point, and the ship will be put around the start point with random heading, waiting the next training epoch. During the training process, the environment conditions (e.g., speed of wind and flow, etc.) is set to change randomly every 20 time steps. To be specific, directions of wind, flow and wave are set to vary between 0 °and 360 °, wind speed is set to vary between 0.1m/s and 1m/s, flow velocity is set to vary between 0.01m/s and 0.1m/s, wave heights is set to vary between 0.003m and 0.04m, wave length is set to vary between 0.01m and 1.5m.

In the reward function, A is set as 10 to provide basic reward. And λ_1 is set to be 5, λ_2 is set to be 5, and $\lambda_3$is set to be 1. Moreover, to avoid system overreaction when a bad action is taken, we further represent the reward function as:

$$R = (R/10) - 1 \tag{16}$$

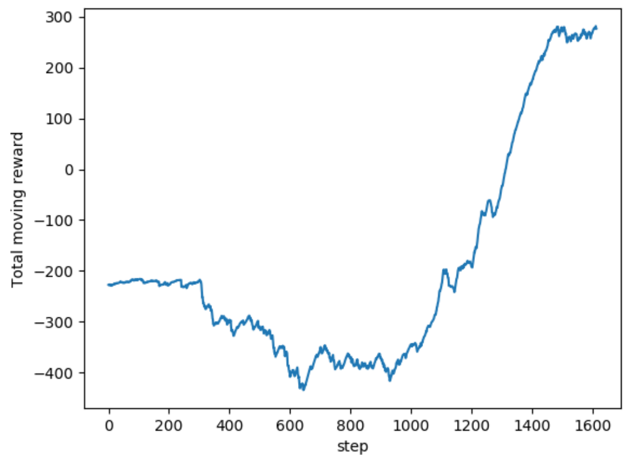


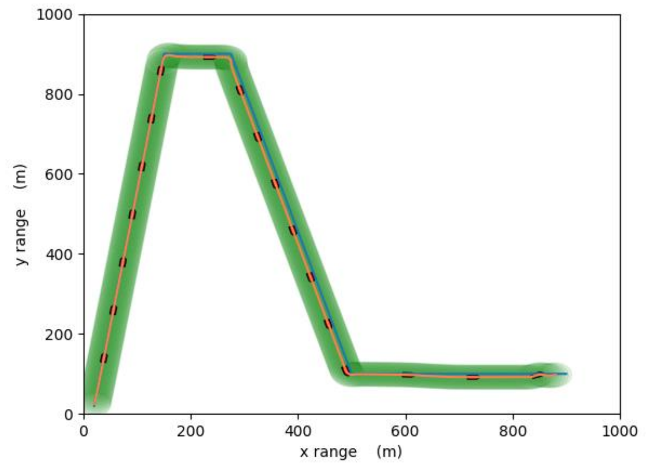
Fig. 6 Moving reward

$$RT_x = 0.99 * RT_{x-1} + 0.01 * epr_x \tag{17}$$

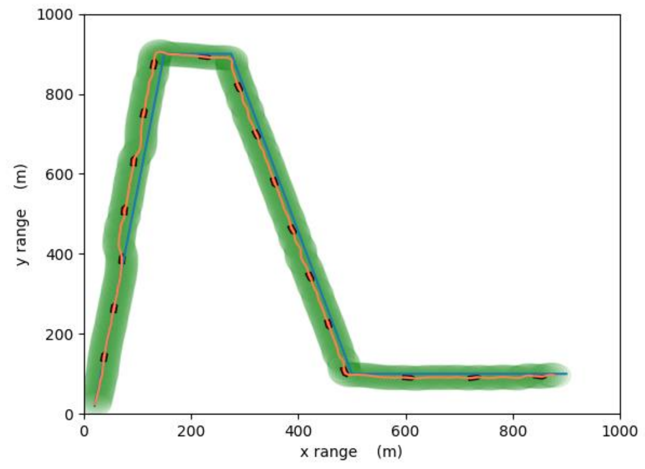
where epr_x denotes the total reward the agent acquires during the xth training epoch. At the beginning, the agent is ignorance of the task, with RT_x continuing to decline. And it fluctuate between 600th and 1000th training epoch, implying that the agents have learned the basic control method. Then it peak at 300 by 1400th epoch.

In the following figures(Fig. 7(a) and Fig. 7(d)), the green area are the max deviation area of the ship. The blue line is the target course, while orange line show the actual trajectory of the ship.

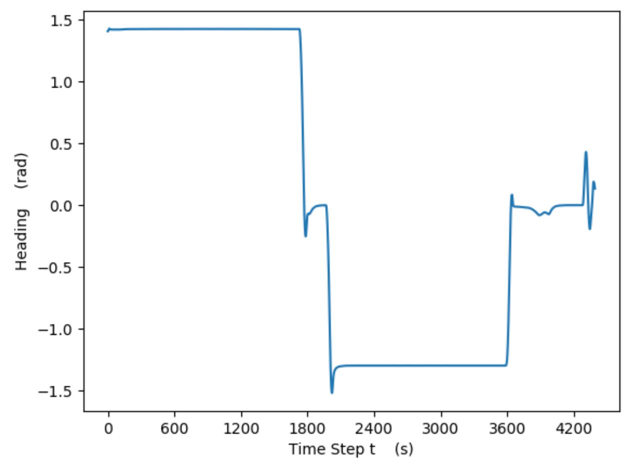
In practice, the disturbance states, especially flow and wave, in a specific sea area show little change during a period of time. The performance in this case (Fig. 7(a)) is stable, with little fluctuation in heading changing. And we also consider some severe environment where the disturbance state keep changing during the whole process (Fig. 7(d)). The difference of performance between stable and changing disturbance can be easily seen in Fig. 7(b) and Fig. 7(e). With constant disturbance, the ship heading is stable during the whole process, while the ship heading keep fluctuating in severe environment. The similar situation are also illustrated in Fig. 7(c) and Fig. 7(f), where the deviations from target course are described. The largest deviations in both situations occur when the target course turns.



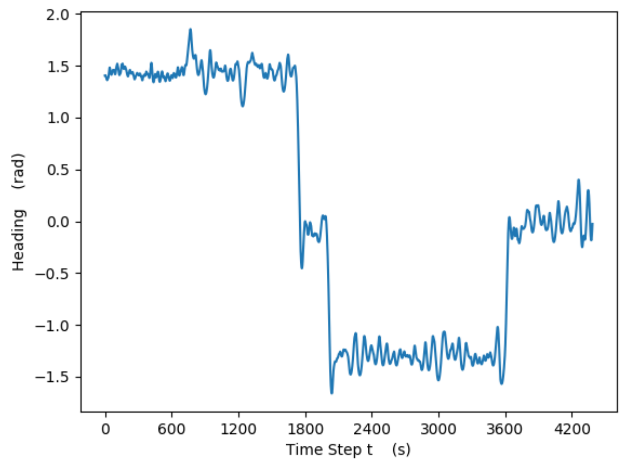
(a) Trajectory with stable disturbance



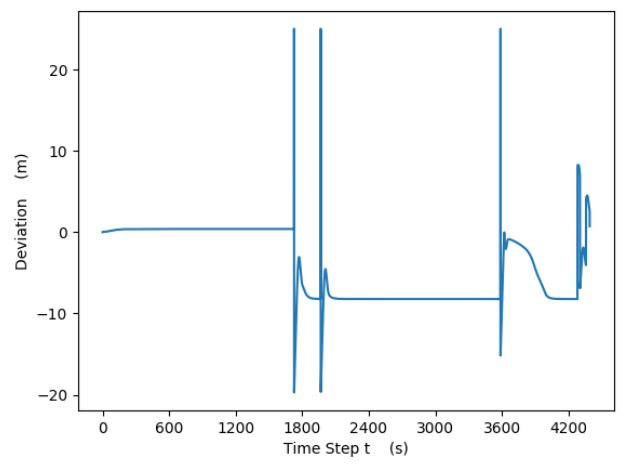
(d) Trajectory with changing disturbance



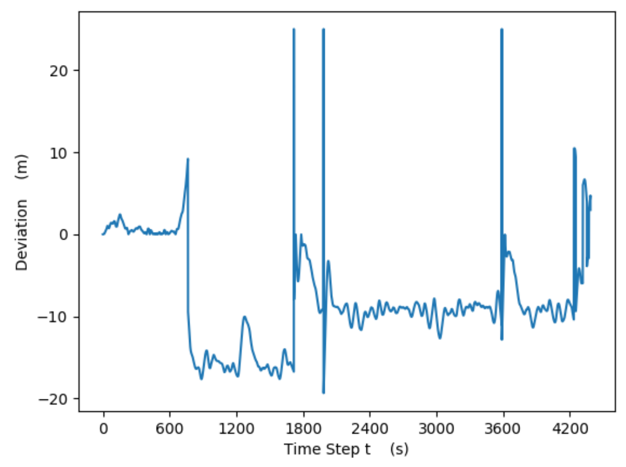
(b) Heading with stable disturbance



(e) Heading with changing disturbance



(c) Deviation with stable disturbance



(f) Deviation with changing disturbance

Fig. 7 Testing results with different kinds of disturbance

CONCLUSION

This paper presents an end-to-end trajectory tracking algorithm for unmanned surface vehicle under complex sea condition using A3C reinforcement learning algorithm. This method do not any prior knowledge and return the control signal directly once the necessary data is given, thus avoiding the cumulative errors derived from each disturbance model. Moreover, it works with low memory requirements and better convergence. Simulation results also reveal that it can adapt to the complex environment quickly. And it can easily be applied to any type of ships by just changing the parameters.

REFERENCES

Abbeel, P., Coates, A., Quigley, M., & Ng, A. Y. (2007). *An application of reinforcement learning to aerobatic helicopter flight.* Paper presented at the Advances in neural information processing systems.

Ashrafiuon, H., Muske, K. R., & McNinch, L. C. (2010). *Review of nonlinear tracking and setpoint control approaches for autonomous underactuated marine vehicles.* Paper presented at the American Control Conference (ACC), 2010.

Bibuli, M., Bruzzone, G., Caccia, M., & Lapierre, L. (2009). Path-Following Algorithms and Experiments for an Unmanned Surface Vehicle. *Journal of Field Robotics, 26*(8), 669-688. doi:10.1002/rob.20303

Brown, N., & Sandholm, T. (2017). *Safe and nested subgame solving for imperfect-information games.* Paper presented at the Advances in Neural Information Processing Systems.

Florence, P. R., Manuelli, L., & Tedrake, R. (2018). Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*.

Fossen, T. I. (2011). *Handbook of marine craft hydrodynamics and motion control*: John Wiley & Sons.

Fossen, T. I. (2012). How to incorporate wind, waves and ocean currents in the marine craft equations of motion. *IFAC Proceedings Volumes, 45*(27), 126-131.

Fossen, T. I., Breivik, M., & Skjetne, R. (2003). Line-of-sight path following of underactuated marine craft. *IFAC Proceedings Volumes, 36*(21), 211-216.

Fossen, T. I., & Lekkas, A. M. (2017). Direct and indirect adaptive integral line ‐ of ‐ sight path ‐ following controllers for marine craft exposed to ocean currents. *International Journal of Adaptive Control and Signal Processing, 31*(4), 445-463.

Fossen, T. I., Pettersen, K. Y., & Galeazzi, R. (2015). Line-of-Sight Path Following for Dubins Paths With Adaptive Sideslip Compensation of Drift Forces. *IEEE Trans. Contr. Sys. Techn., 23*(2), 820-827.

Ihle, I. A. F., Arcak, M., & Fossen, T. I. (2007). Passivity-based designs for synchronized path-following. *Automatica, 43*(9), 1508-1518. doi:10.1016/j.automatica.2007.02.018

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., . . . Kavukcuoglu, K. (2016). *Asynchronous methods for deep reinforcement learning.* Paper presented at the International conference on machine learning.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

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

Moreira, L., Fossen, T. I., & Soares, C. G. (2007). Path following control system for a tanker ship model. *Ocean Engineering, 34*(14), 2074-2085.

Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., . . . Petersen, S. (2015). Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., . . . Lanctot, M. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., . . . Graepel, T. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., . . . Bolton, A. (2017). Mastering the game of Go without human knowledge. *Nature, 550*(7676), 354.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*: MIT press.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., . . . Vanhoucke, V. (2018). Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. *arXiv preprint arXiv:1804.10332*.

Wang, N., Gao, Y., Sun, Z., & Zheng, Z. (2018). Nussbaum-based adaptive fuzzy tracking control of unmanned surface vehicles with fully unknown dynamics and complex input nonlinearities. *International Journal of Fuzzy Systems, 20*(1), 259-268.

Xiang, X., Yu, C., Lapierre, L., Zhang, J., & Zhang, Q. (2018). Survey on fuzzy-logic-based guidance and control of marine surface vehicles and underwater vehicles. *International Journal of Fuzzy Systems, 20*(2), 572-586.