Model-Free Recurrent Reinforcement Learning for AUV Horizontal Control

Yujia Huo[1,2,a], Yiping Li[1,b] and Xisheng Feng[1]

[1]State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, 110016, China
[2]University of Chinese Academy of Sciences, Beijing, 100049, China

Corresponding e-mail: [a]huoyujia@sia.cn; [b]lyp@sia.cn

Abstract. In this paper, aiming at the problems of 2-DOF horizontal motion control with high precision for autonomous underwater vehicle(AUV) trajectory tracking tasks, deep reinforcement learning controllers are applied to these conditions. These control problems are considered as a POMDP (Partially Observable Markov Decision Process). Model-free reinforcement learning(RL) algorithms for continuous control mission based on Deterministic Policy Gradient(DPG) allows robots learn from received delayed rewards when interacting with environments. Recurrent neural networks LSTM (Long Short-Term Memory) are involved into the reinforcement learning algorithm. Through this deep reinforcement learning algorithm, AUVs learn from sequences of dynamic information. The horizontal trajectory tracking tasks are described by LOS method and the motion control are idealized as a SISO model. Tanh-estimators are presented as data normalization. Moreover, AUV horizontal trajectory tracking and motion control simulation results demonstrate this algorithm gets better accuracy compared with the PID method and other non-recurrent methods. Efforts show the efficiency and effectiveness of the improved deep reinforcement learning algorithm.

1. Introduction

Benefiting from the continuous advances in several fields like control, computer, sensor and communication, autonomous underwater vehicles(AUVs) have play irreplaceable rolls in maritime application both in civilian and military areas including oceanographic survey, underwater operation and maritime reconnaissance and surveillance. The autonomy of AUVs has been one of the most critical criteria and been studied for decades by various control theories ranging from traditional techniques to several different artificial neural network-based control architectures.

Conventional approaches focus on controlling an AUV described by one or limited and very few accurate models with hydrodynamic parameters obtained from experiments and achieve variable results. Due to the inertial, buoyancy and hydrodynamic effects, dynamics of AUVs are with strong nonlinearity. AUVs suffer from the surrounding disturbance and the uncertainty when operating underwater. Linear approximations of robot dynamic model are insufficient and nonlinear models are destined for inaccuracy because some parameters are unknown or vary with un-modelled conditions.

With the rapid developments of both computing hardware and artificial intelligence in recent years, artificial agents with learning abilities achieve encouraging performance particularly in the condition of making plans and decisions for complex systems which difficult to establish a set of accurate models. Reinforcement learning(RL) allows agents learn the action when interacting with environment so as to maximize some notion of cumulative reward. In [1], reinforcement learning was applied to an AUV. A Q(λ)-learning algorithm for 3D-navigation control of an AUV URIS[2] and a semi on-line-neural Q-learning(SONQL) in [3] were proposed. For enhancing and accelerating learning in real robotics applications, a tow-step structure actor-critic(AC) reinforcement learning was designed for underwater robotic behaver learning. An open-frame AUV Ictineu was trained on simulation platform approximated to the environment and the experience of policy from simulation remained when agents continuing training in the real world[4]. Adaptive RL was integrated into the adaptive control algorithm and rigorous theoretical analysis is proposed to prove the stability[5]. In other similar motion control application of the UAV including multirotors and helicopters[6,7], RL was also involved.

To improve the performance of agents, deep neural networks with RL has made significant achievements in discrete state/action space, for example playing Atari with the method deep q-networks(DQN)[8]. A algorithm with deterministic policy gradient(DPG) instead of ineffective stochastic policy gradient was proposed in [9] and based on this algorithm, deep deterministic policy gradient(DDPG) was introduced in later work[10]. Application on AUV depth control[11] and heading control[12], DDPG was introduced and compared with PID controller[12], linear quadratic Gaussian integral controller[11] and nonlinear model predictive controller[11].

However, all of these works based on model-free reinforcement learning assume fully observed state[13]. Due to the sample frequency deference and noise of sensors, unobserved variations of systems under control, or state-aliasing caused by function approximation. Systems with these problems are described as Partially-Observable Markov Decision Processes(POMDP). In [14], a RL algorithm with a LSTM recurrent neural network was presented and solve non-Markovian tasks. In [15], recurrent networks replaced the DQN's first fully connected layer [8] and a better performance was got. Multiple LSTM layers approximating the Q-function controlled a real 7-axis articulated robot arm for high precision assembly tasks[16]. [13] presented an algorithm allows the agent to solve POMDP in continuous state/action space primarily considering the DPG and this algorithm gets an effective and satisfying result.

This paper aims to design a controller based on recurrent reinforcement learning for AUV trajectory tracking tasks on horizontal plane when these problems are considered as a POMDP.

The remainder of the paper is organized as follows. In section 2, the underwater dynamic system description is presented. Section 3 explains the composition of controller combining LSTM deterministic policy gradient algorithm. In section 4, simulation study is provided the effectiveness of the algorithm proposed, followed by conclusion in section 5.

2. Problem Formulation

AUVs operate underwater with six degrees of freedoms in three dimensional space but the system with coupled dynamics is difficult to control in practice works. To simplify the controller design, a motion task in 3D space is often decomposed in speed, steering and depth control.

In this paper, we consider the AUV moves and track a desire trajectory in the horizontal plane. Figure 1 illustrates the motion of the AUV in the horizontal plane.

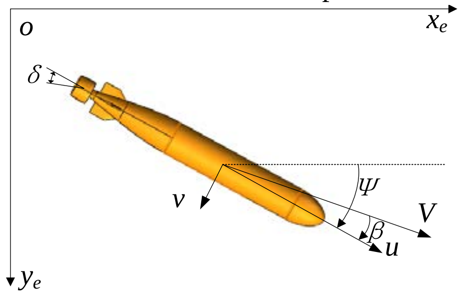

Figure 1. AUV motion in horizontal plane

In Figure 1, a positon coordinate frame $\{OX_eY_e\}$ is defined as the earth-fixed coordinate in the top view and accords to the right-hand system. *V* is the velocity of the AUV and two decoupled velocities *u* in surge and *v* in sway. Heading angel Ψ is defined as the angle between the axis X_e and the nose pointing of AUV. Drift angle is defined as the angle between the velocity *V* and the velocity *u* in surge. δ is the angel of the vertical rudder which controls the heading of the robot.

According to [17], an AUV horizontal plane dynamics can be described as follows:

$$\begin{bmatrix} m-Y_{\dot{v}_r} & -Y_{\dot{r}} & 0 \\ -N_{\dot{v}_r} & I_{zz}-N_{\dot{r}} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \dot{v} \\ \dot{r} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} Y_{uv}u & (Y_r-m)u & 0 \\ N_{uv}u & N_r u & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} v \\ r \\ \psi \end{bmatrix} + \begin{bmatrix} Y_\delta u^2 \\ N_\delta u^2 \\ 0 \end{bmatrix}\delta + \begin{bmatrix} Y_{r|r|}r|r| + Y_{v|v|}v|v| \\ N_{r|r|}r|r| + N_{|v|r}|v|r + N_{|v|v}|v|v \\ 0 \end{bmatrix} \tag{1}$$

Where we consider the center of gravity $[x_G, y_G, z_G]^T$ stays at the body-fixed vehicle coordinate system origin and same as the geometric center of the AUV. Then we find it convenient to assume that the sway *v* is far less than the surge *u* and a simplified discrete-time dynamic model for heading control in the horizontal plane can be expressed as follows:

$$\begin{bmatrix} r(k+1) \\ \psi(k+1) \end{bmatrix} = T_s M^{-1}\left[\left(\begin{bmatrix} N_r u & 0 \\ 1 & 0 \end{bmatrix} + \frac{M}{T_s}\right)\begin{bmatrix} r(k) \\ \psi(k) \end{bmatrix} + \begin{bmatrix} N_\delta u^2 \\ 0 \end{bmatrix}\delta + \begin{bmatrix} N_{r|r|}r|r| + N_{|v|r}|v|r + N_{|v|v}|v|v \\ 0 \end{bmatrix}\right] \tag{2}$$

Where *M* is the is

$$\begin{bmatrix} I_{zz}-N_{\dot{r}} & 0 \\ 0 & 1 \end{bmatrix} \tag{3}$$

And T_s is the sampling time. Thus, a dynamic model function which describe a map from action rudder angle δ to states yaw *r* and heading angle ψ.

In trajectory tracking tasks, line-of-sight method is employed to detect a target position along the desire trajectory. When operating underwater, the heading angle ψ_{AUV}, the target position (x_{target}, y_{target}) and the actual position (x_{AUV}, y_{AUV}) are known. And

$$x_t = x_{AUV} - x_{target}; \quad y_t = y_{AUV} - y_{target} \tag{4}$$

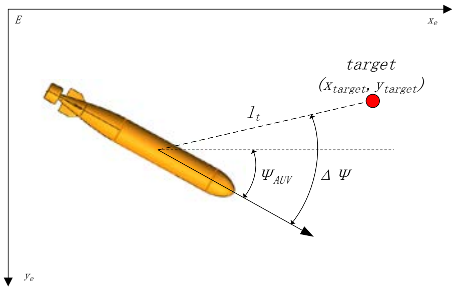

Figure 2. Line of sight(LOS) guidance approach

Figure 2 illustrates the line-of-sight guidance approach. Where $\Delta\Psi$ means the angle between AUV heading angle and the angle towards the target position. Now we have

$$l_t = \sqrt{x_t^2 + y_t^2} = \sqrt{(x_{AUV} - x_{target})^2 + (y_{AUV} - y_{target})^2}; \quad \Delta\psi = \psi_{AUV} - \psi_{target} = \psi_{AUV} - \arctan\left(\frac{y_{AUV} - y_{target}}{x_{AUV} - x_{target}}\right) \tag{5}$$

Thus, the target position can be describe as $(l_t, \Delta\psi)$ in a polar coordinate.

As the problem of control in a 3D space has been separated into a steering control, diving control and speed control. For the control problem in the horizontal plane, we only consider the relationship between the heading angle rate *r* , its accumulation form - heading angle Ψ and the action rudder angle δ. The velocity in surge *u* is often set as a constant. The desire trajectory is composed of a sequence of target spots $(x^1_{target}, y^1_{target}, x^2_{target}, y^2_{target}, \cdots)$ and these target spots are equidistant along with the trajectory.

3. LSTM Deterministic Policy Gradient

As mentioned in introduction, reinforcement learning has produced several achievements but all these works assume fully observe state. In this section, a deep reinforcement learning algorithm with current neural networks is introduced. Since the algorithm has an actor-critic structure based on DPG[18] with an actor neural network and a critic neural network synonyms for the policy and value function. An off-policy deterministic actor-critic method is also presented in this section.

3.1 Partially Observable Markov Decision Process

A Markov decision process(MDP) is a stochastic process which satisfies the Markov property. When interacting with the environment at each of a sequence of discrete time steps, $t = 0,1,2,......$, the agent receives representation of the environmental state, $s_t \in \mathcal{S}$, where $\mathcal{S}$ is the set of possible states, and on that basis chooses an action, $a_t \in \mathcal{A}\ (S_t)$,where $\mathcal{A}\ (s_t)$ is the set of actions available in the state s_t [19]. At each step after action, consequently, the agent receives a numerical reward, $r_{t+1} \in \mathcal{R} \subset \mathbb{R}$, and transfer itself in new state s_{t+1}. Thus a MDP is described (s_t, a_t, r_t,, $p(s_{t+1}=s'|s_t=s,\ a_t=a)$), where $p(s_{t+1}=s'|s_t=s,\ a_t=a)$ is transition probability that action a in state s at time t will lead to state s' at time $t+1$.

But in the real world, the full states of the system being needed are rarely provided to the agent or even perceived by sensors. In other words, it is rare that the Markov property holds well in the real world. A Partially Observable Markov Decision Process(POMDP) captures the environmental dynamics of the real world more effectively by explicitly acknowledging that the sensations that the agent receives are only partial glimpses of the underlying system state[15]. Then, a POMDP is described as (s, a, p, r, Ω, O) with 6 components, and the s, a, p, r are described as states, actions, transitions and rewards as before. But the agent no longer receives the states directly from the system, instead an observation $o \in \Omega$ is received. And this observation is based on the probability distribution $o \sim O(s)$.

3.2 Off-Policy Deterministic Actor-Critic

For a MDP, the performance can be described and evaluated as an expectation with the conditional probability density at $\pi_\theta(s_t,\ a_t)$ as following:

$$J(\pi_\theta) = \int_S \rho^\pi(s) \int_A \pi_\theta(s,a) r(s,a) \mathrm{d}a \mathrm{d}s = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[r(s,a)] \tag{6}$$

where we denote the discounted state distribution by

$$\rho^\pi(s') = \int_S \sum_{t=1}^{\infty} \gamma^{t-1} p_1(s) p(s \to s', t, \pi) \mathrm{d}s \tag{7}$$

For this performance function(6), the basic goal is to maximize the performance function with the optimal policy. And the policy gradient is widely utilized for a continuous state/action tasks. By updating the parameters θ with the performance gradient(8), the algorithm finds the optima parameterized with θ, and at the same time, maximizing the time cumulative reward function.

$$\nabla_\theta J(\pi_\theta) = \int_S \rho^\pi(s) \int_A \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) \mathrm{d}a \mathrm{d}s = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)] \tag{8}$$

$$\theta \leftarrow \theta + \alpha \widehat{\nabla_\theta J(\theta)} \tag{9}$$

where $\alpha \widehat{\nabla_\theta J(\theta)}$ is a stochastic approximation of the true gradient of the performance function and $Q^\pi(s,a)$ is the action-value function but not easy to be estimated.

As an extended method based on the policy gradient, the actor-critic has a better performance. The actor-critic contains two eponymous components. The *actor* adjusts the stochastic policy $\pi_\theta(s)$ with parameters θ by stochastic gradient of the performance function(8). Meanwhile, as the action-value function $Q^\pi(s,a)$ hard to estimate, an approximate policy evaluation algorithm such as a temporal-difference(TD) method is introduced to *critic* and estimate the action-value function $Q^w(s,a) \approx Q^\pi(s,a)$ with parameters ω.

To improve the actor-critic algorithm, a deterministic policy replaces the stochastic policy, because the gradients of deterministic policy can be estimated more efficiently without a problematic integral over the action space [18].And it has been proved that the deterministic policy gradient is a limiting case of the stochastic policy gradient theorem.

Unlike the stochastic policy as mentioned before, the actor updates the parameter θ of the deterministic policy $\mu_\theta(s)$. Since the we choose a deterministic policy, an off-policy algorithm is needed to allow the agent to explore and consequently to learn the evaluation. Then, the updating for an off-policy deterministic actor-critic with an off-policy algorithm Q-learning in *critic* and a deterministic policy gradient in *actor* can be described as following:

$$\delta_t = r_t + \gamma Q^w(s_{t+1}, \mu_\theta(s_{t+1})) - Q^w(s_t, a_t) \tag{10}$$

$$w_{t+1} = w_t + \alpha_{critic} \delta_t \nabla_w Q^w(s_t, a_t) \tag{11}$$

$$\theta_{t+1} = \theta_t + \alpha_{actor} \nabla_\theta \mu_\theta(s_t) \nabla_a Q^w(s_t, a_t) \Big|_{a=\mu_\theta(s)} \tag{12}$$

Where, δ_t is the TD error of Q-learning used for critic.

3.3 LSTM Deterministic Policy Gradient

As mentioned above, the states cannot be received by the agent. Instead, the agent only indirectly observes the Markov decision process through the observations. Then there involves a history $h_t = (o_1, a_1, o_2, a_2, \cdots, o_{t-1}, a_{t-1}, o_t)$. Same as the goal of off-policy deterministic actor-critic algorithm, we tend to maximize the deterministic policy performance function(13) re-described for POMDP.

$$J = \mathbb{E}_\tau[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_{t,} \mu(h_t))] \tag{13}$$

In the case of partial observability, the optimal policy and the estimated action-value function are both considering the observation-action h_t with states as a sequence of history, instead of just states of one step for fully observable MDP. To solve the POMDP, feedforward networks policy and evaluation in DPG are replaced with neural networks(LSTM) in practice, which allow the agent to learn from the information preserved from past. Thus, a new policy gradient is obtain as following by replacing $\mu(s)$ and *Q(s, a)* with *(h)* and *Q(h, a)*.

$$\nabla_\theta J = \mathbb{E}_\tau[\sum_t \gamma^{t-1} \frac{\partial Q^\mu(h_t, a)}{\partial a}\Big|_{a=\mu^\theta(h_t)} \frac{\partial \mu^\theta(h_t)}{\partial \theta}] \tag{14}$$

where trajectory $\tau = (s_1, o_1, a_{1,} s_2, o_2, a_2 \cdots)$.

Based on ideas from the DQN[8] method, a relay buffer is also introduced which can improve the data efficiency and stability. Meanwhile, target networks are appended to the algorithm. Two copies of the evaluation function Q and the policy are involved, with parameters *w'* and θ'. *w'* and θ' are updated as mirror of *w* and θ with some delay. This Asynchronous update also enhances the stability of the neural networks.

LSTMDPG algorithm:

```
Initialize critic network $Q^w(a_t, h_t)$ and actor $\mu^\theta(h_t)$ with parameters $w$ and $\theta$
Initialize target network $Q^{w'}(a_t, h_t)$ and $\mu^{\theta'}(h_t)$ with weight $w' \leftarrow w$ and $\theta' \leftarrow \theta$
Initialize replay buffer R
for episode i=1 to max episode M do:
  Initialize empty history memory $h_0$
  for j=1 to T do:
    Observe $o_t$
    Append observation and action to previous history memory $h_t \leftarrow h_{t-1}, a_{t-1}, o_t$
    Choose action $a_t = \mu(h_t \mid \theta)$
  end for
  Store the sequence $(o_1, a_1, r_1, \cdots, o_T, a_T, r_T)$ to R
  Sample a minibatch $(o_1^n, a_1^n, r_1^n, \cdots, o_T^n, a_T^n, r_T^n)_{n=1\sim N}$ with shape N episodes from R
  Update $h_t^n = (o_1^n, a_1^n, \cdots, o_{t-1}^n, a_{t-1}^n, o_t^n)$
```

```
    Compute target values $(\delta_t^n, \cdots, \delta_T^n)$ for each sample episode by LSTM
        $\delta_t^n = r_t^n + \gamma Q^{w'}(h_{t+1}^n, \mu^{\theta'}(h_{t+1}^n)) - Q^w(h_t^n, a_t^n)$
    Calculate critic network and actor network update with BPTT[20]
        $\Delta w = \frac{1}{NT}\sum_n\sum_t \delta_t^n \frac{\partial Q^w(h_t^n, a_t^n)}{\partial w}$
        $\Delta \theta = \frac{1}{NT}\sum_n\sum_t \delta_t^n \frac{\partial Q^w(h_t^n, \mu^\theta(h_t^n))}{\partial a}\frac{\partial \mu^\theta(h_t^n)}{\partial \theta}$
    Update actor and critic with Adam[21]
    Update the target networks with the learning rate $\alpha_{critic}$ and $\alpha_{actor}$
        $w' \leftarrow \alpha_{critic} w + (1-\alpha_{critic}) w'$
        $\theta' \leftarrow \alpha_{actor} \theta + (1-\alpha_{actor}) \theta'$
end for
```

4. Simulation

In this section, results of simulation for the AUV trajectory tracking tasks are presented. Primary parameters of AUV in simulation are listed. The simulation platform is coded in Python3.5 on Ubuntu16.04 system and TensorFlow is in use for deep neural networks implementation.

In comparison with the LSTM deterministic policy gradient(LSTMDPG), a PID controller and a simple deep deterministic policy gradient(DDPG)[10] controller are designed for simulation of AUV trajectory tasks.

In LSTMDPG algorithm, both the actor network and critic network are LSTM recurrent neural networks with 4 layers and the number of hidden layer units is 200. In the practice, we set the discount factor γ as 0.99, learning rate α as 0.001 and the batch size as 64.

In DDPG algorithm, as the problem we aim to solve is in a continuous state/action space, it is enough that both the actor network and the critic network are fully connected. The numbers of the units in the hidden layers of both actor and critic networks are 64. And discount factor γ, learning rate α, the batch size and the activation function of the neural networks are each set as 0.99, 0.001, 64 and Relu.

It is appropriate to define the input as an error form like (7). Both in the LSTMDPG and DDPG, the action δ and state $\Delta\psi$, l and r are normalized by the tanh-estimators which are robust and highly efficient. The normalization is given by:

$$x_k' = \tanh\left(\frac{x_k - \mu}{\sigma}\right) \tag{15}$$

where, μ and σ are the mean and standard deviation estimates. Standard deviations of states $\Delta\psi \in$ (-180°,180°], r and the action $\delta \in$ [-45°,45°] that transferred into neural networks for training are each set as $\sigma_{\Delta\psi} = 80; \sigma_r = 4; \sigma_\delta = 20, \sigma_l = 10$, and the normalizations are shown in figure 3.

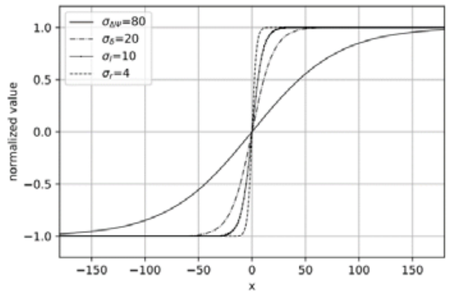

Figure 3. The normalization of states and action

Then for reinforcement learning of the trajectory tracking tasks on the horizontal plane, the reward that agent receives at each step is set as the quadratic sum of the states and actions with different weights:

$$reward(k) = -[\Delta\psi(k)^2 + r(k)^2 + l(k)^2 + 0.1 * \delta(k)^2] \tag{16}$$

And during training the AUV, the states of the agent are reset as ($\Delta\psi$=45°+*dis*, *l =10*, *r=0*) at the beginning of each episode, where *dis* is a random disturbance.

For another comparison, a PID controller is also designed and the controller can be described as following and the factors of PID are set as (0.6,1.5,0.08):

$$\delta(t) = K_p \Delta\psi(t) + K_d \frac{\mathrm{d}\Delta\psi(t)}{\mathrm{d}t} + K_I \int_0^t \Delta\psi(\tau)\mathrm{d}\tau \tag{17}$$

Random disturbances appended to the observation are described by a Gaussian distribution *N(0,0.05).*

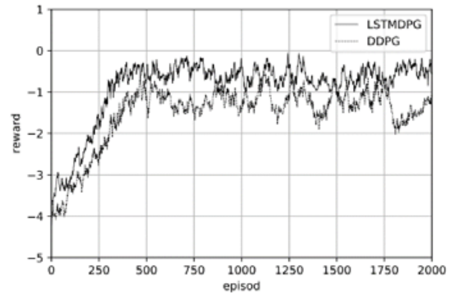

Figure 4. Rewards of LSTMMPG and DDPG.

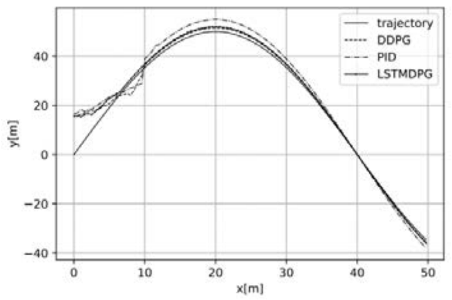

Figure 5. Trajectory tracking with different algorithm.

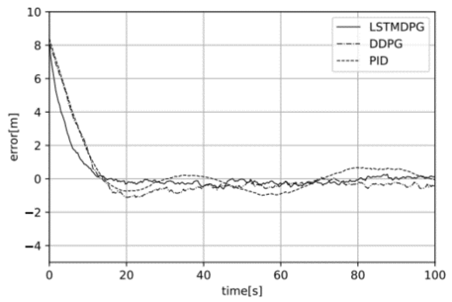

Figure 6. Error of trajectory tracking with different algorithm.

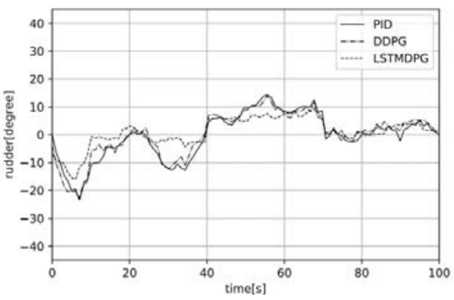

Figure 7. Rudder action of different algorithm.

We set a maximum of 2000 episodes to train both DDPG and LSTMDPG, and the rewards in figure 4 illustrate LSTMDPG has accelerated convergence speed. Furthermore, the LSTMDPG has a better performance with bigger rewards. Figure 5 and figure 6 show the trajectory and the error on the horizontal plane when agents are operating in the tracking task. It can be seen that LSTMDPG has a fast convergence with little steady-state error when compared with a DDPG controller and a PID controller. And figure 7 is shows the rudder action of the AUV.

Simulations illustrate that a LSTMDPG controller can solve the trajectory tracking tasks with a satisfying performance which is assumed as a POMDP

5. Conclusion

In this paper, a trajectory tracking task assumed as a partially observable Markov decision process is solved by a recurrent reinforcement learning. Extended from deterministic policy gradient, LSTM recurrent neural networks are involved and replace the feedforward networks both in actor and critic. This method with recurrent neural networks allow agent to learn effectively, and is validated in simulation compared with a PID controller and DDPG controller.

References

[1] Gaskett C 1999 Reinforcement Learning applied to the control of an Autonomous Underwater Vehicle *Proc. Aust. Conf. Robot. Autom.*
[2] Carreras M, Batlle J and Ridao P 2001 Hybrid coordination of reinforcement learning-based behaviors for AUV control *Proc. 2001 IEEE/RSJ Int. Conf. Intell. Robot. Syst. Expand. Soc. Role Robot. Next Millenn. (Cat. No.01CH37180)* **3** 1410–5
[3] Carreras M, Yuh J, Batlle J and Ridao P 2005 A Behavior-Based Scheme Using Reinforcement Learning for Autonomous Underwater Vehicles *IEEE J. Ocean. Eng.* **30**
[4] El-Fakdi A and Carreras M 2013 Two-step gradient-based reinforcement learning for underwater robotics behavior learning *Rob. Auton. Syst.* **61** 271–82
[5] Cui R, Yang C, Member S S S, Li Y, Member S S S, Sharma S, Member S S S, Li Y and Member S S S 2017 Adaptive Neural Network Control of AUVs With Control Input Nonlinearities Using Reinforcement Learning *Ieee Trans. Syst. Man, Cybern. Syst.* 1–11
[6] Bou-Ammar H, Voos H and Ertel W 2010 Controller design for quadrotor UAVs using reinforcement learning *Proceedings of the IEEE International Conference on Control Applications* pp 2130–5
[7] Ng A Y, Coates A, Diel M, Ganapathi V, Schulte J, Tse B, Berger E and Liang E 2006 Autonomous inverted helicopter flight via reinforcement earning *Springer Tracts Adv. Robot.* **21** 363–72
[8] Mnih V, Silver D and Riedmiller M Playing Atari with Deep Reinforcement Learning *NIPS* 1–9
[9] Conditions A R 2014 Deterministic Policy Gradient Algorithms : Supplementary Material *Icml* **1**
[10] Lillicrap T P, Hunt J J, Pritzel A, Heess N, Erez T, Tassa Y, Silver D and Wierstra D Continuous control with deep reinforcement learning *arXiv Prepr. arXiv1509.02971, 2015.*
[11] Wu H, Song S, You K and Wu C 2018 Depth Control of Model-Free AUVs via Reinforcement Learning *IEEE Trans. Syst. Man, Cybern. Syst.* 1–12
[12] Yu R, Shi Z, Huang C, Li T and Ma Q 2017 Deep Reinforcement Learning Based Optimal Trajectory Tracking Control of Autonomous Underwater Vehicle *36th Chinese Control Conf.* 4958–65
[13] Heess N, Hunt J J, Lillicrap T P and Silver D 2015 Memory-based control with recurrent neural networks *NIPS Deep Reinf. Learn. Work.* 1–11
[14] Bakker B 2003 Reinforcement learning with long short-term memory *In Advances in Neural Information Processing Systems 15 (NIPS-2002* pp 1475–82
[15] Hausknecht M and Stone P 2015 Deep Recurrent Q-Learning for Partially Observable MDPs 29–37
[16] Inoue T, De Magistris G, Munawar A, Yokoya T and Tachibana R 2017 Deep Reinforcement Learning for High Precision Assembly Tasks
[17] Feldman J 1979 DTNSRDC Revised Standarrd Submarine Equations of Motion 12
[18] Silver D, Lever G, Heess N, Degris T, Wierstra D and Riedmiller M 2014 Deterministic Policy Gradient Algorithms *Proc. 31st Int. Conf. Mach. Learn.* 387–95
[19] Sutton R S and Barto A G 1998 *Reinforcement learning: an introduction.* vol 9
[20] Werbos P J 1990 Backpropagation Through Time: What It Does and How to Do It *Proc. IEEE* **78** 1550–60
[21] Kingma D P and Ba J Adam: A Method for Stochastic Optimization *arXiv Prepr. arXiv1412.6980, 2014.*