Deep Reinforcement Learning Algorithms for Ship Navigation in Restricted Waters

Jonathas Marcelo Pereira Figueiredo, Rodrigo Pereira Abou Rejaili

Abstract – Reinforcement Learning has not been fully explored for the automated control of ships maneuvering movements in restricted waters. Nevertheless, more robust and efficient control can be achieved with such algorithms. This paper presents the use of Deep Q Network and Deep Deterministic Policy Gradient methods with a numerical simulator for ship maneuvers to develop control laws. Both methods proved to be efficient in navigational control through a channel. A comparison of response and control behavior resulting from each of the methods is presented.

Keywords – Reinforcement Learning; Navigation; Neural Networks; Deep Learning

1 Introduction

Automation is being used more frequently to augment the efficiency of various systems and processes. One of its objectives is to reduce or eliminate human intervention completely in critical processes where errors are not tolerated. The navigation control for means of transportation is an example where the automation is intended to reduce accidents caused by human influence on piloting and control.

Recently, Artificial Neural Networks (ANNs) and other Machine Learning methods have presented good results in automation of those systems. Cars, drones and helicopters are some examples where those methods have been successful (GERLA ; et al., 2014), (CUTLER; HOW, 2015). On the other hand, there are still some problems where definite automation solutions have not been entirely developed and applied, such as the ship navigation control in restricted waters.

Nowadays, such tasks are done by a ship commander who is specialized in this kind of navigation. The ship control is based on his/her knowledge of the environmental/meteorological conditions and experience in berthing and transport in the port region, thus demanding experienced professionals, specifically for each region.

Nevertheless, this process still presents human risks which are the recurrent cause of maritime accidents (HETHERINGTON; FLIN; MEARNS, 2006). Thus, the automation of such processes is an alternative to reduce the risks associated with the maneuvering process in restricted waters. This subject is still insufficiently explored, being a pertinent research subject for the nautical sector (AHMED; HASEGAWA, 2014).

(A ser preenchido pelos editores) Versão inicial submetida em dd de mmm. de 2017. Versão final aceita em dd de mmm. de 2017. Publicado em dd de mmm. de 2017. Digital Object Identifier _____/_____

1.1 State of the art

Neural Networks have been applied to transport systems in general since the 1990s, as in the work of (DOUGHERTY, 1995). Some control systems have already used Reinforcement Learning (RL) methods successfully, and it has been demonstrated how some adaptations are viable to conceive controllers using RL (HAFNER; RIEDMILLER, 2011). Nevertheless, their application in the automation of ship maneuvering is still scarcely explored in scientific literature (AMENDOLA ; et al., 2018). One exception is the ANN-based controller for ship berthing assuming known navigation trajectories (AHMED; HASEGAWA, 2014).

The use of Reinforcement Learning algorithms for those tasks has only begun in the last decades. One of the first relevant results found in the literature is an experiment using an artificial neuron actor-critic agent and a simulation of the ship navigating through a channel using sensors that measure the course, the angle between the course and the closest group of buoys, and the distance to this group (STAMENKOVICH, 1992). A comparison between the application of SARSA and Q-learning algorithms with a discrete state model to control the attack angle of the ship, in navigation on restricted waters with constant speed and small obstacles (LACKI, 2008). More recently, Q-learning (on-line) using discrete states was compared with the Least Squares Policy Iteration (LSPI) for continuous states using function approximators (off-line) (RAK; GIERUSZ, 2012). In this study, the authors sought to generate the navigation trajectory using RL, from a certain configuration of the channel (obstacle placement) that had a final position as goal. A RL strategy was proposed using Fitted Q-Iteration with batchs of simulated experience generated with the *Tanque de Provas Numérico* (TPN) simulator, through discrete control actions and using variable speed, aiming to follow a guideline in a channel (AMENDOLA ; et al., 2018). The obtained results, however, were not satisfactory and the ship had oscillatory movements around the guideline following the navigation policy learned with the algorithm.

1.2 Objectives

This paper aims to apply ANNs together with RL algorithms, using the TPN maneuver simulator (FILHO; ZIMBRES; TANNURI, 2014) to develop a steering and propulsion control system for a ship. The performance of such algorithms is then evaluated based on its ability to navigate a ship following a specific trajectory.

2 Methods

This work proposes the comparison of two trajectory controllers learned with RL models using two methods: Deep Q Network (DQN) (MNIH ; et al., 2015) and Deep Deterministic Policy Gradient (DDPG) (LILLICRAP ; et al., 2015). Both methods are briefly explained in the following sections.

2.1 Deep Q Network

Deep Q Network is an architecture originally proposed to learn how to play 49 classical Atari 2600 games, using a state observation from the agent solely through the game visualization (pixels from the game screen). For such, the authors approximate the action value function (Q) using a Convolutional Neural Network (CNN) that receives the state observation as the input and returns the action-values as outputs. The authors also proposed a training routine using experience replay, i.e. using random experimental transition samples to train the CNN, eliminating the correlation in the sequence of observations (which affects the convergence of

the CNN) and smoothing the changes in the data distribution. Therefore, during training, the updates of the Q-learning are made with samples (or minibatches) of experimental tuples (s, a, r, s'), picked randomly from the samples stored in $U(D)$ (replay). The weights (θ) of the Q network are updated with the following loss function (for the i-th step):

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right]; \tag{1}$$

where:

- θ_i are the Q Network parameters on the i-th step;
- θ_i^- are the parameters of the ANN used to compute the target - $\max_{a'} Q(s', a'; \theta_i^-)$ - on the i-th step.

The ANN is trained using Adam (KINGMA; BA, 2014). The target network parameters (θ_i^-) are only updated with the Q Network parameters (θ_i) each C steps, remaining constant between individual updates, what the authors call Hard Target Model Update. Another option, the Soft Target Model Update, updates the parameters at every step of the algorithm, through a weighted sum of the old model and the new one following the next equation (C defines the weights of the update):

$$\theta_i^- = C \times \theta_i^- + (1 - C) \times \theta_i. \tag{2}$$

Even though DQN has good results in high dimensional systems, its action space is discrete. Many interest tasks, especially control ones (including the one analyzed in this paper), have a continuous action space. If the action space is discretized too finely, the action space becomes excessively large and the problem complexity rises, hindering the method convergence.

2.2 Deep Deterministic Policy Gradient

The Deep Deterministic Policy Gradient (DDPG) method is usually applied to environments where the action space is continuous, as the rudder and propeller action control in navigation. Such method is based on the actor-critic architecture, which is used to represent the policy function independently from the value function. The policy function structure is called actor and the value function structure, critic. The actor produces an action, given the current environment state, and the critic produces a TD (Temporal Differences) error signal, given the state and resulting reward. The critic output boosts both the actor and critic learning.

In the DDPG, the policy algorithm used is $a = \mu_\theta(s_t|\theta^\mu)$. This means that instead of using $\pi_\theta(a|s) = P(a|s, \theta)$ as in the stochastic cases, a deterministic policy is chosen. This way the gradient is calculated only over the action space, which requires a smaller amount of samples than in the stochastic case. Nonetheless, a deterministic policy does not explore entirely the states space, thus, to overcome this limitation, a noise process is added $(\mathcal{N}_t)$. The policy then becomes the following:

$$a = \mu_\theta(s_t|\theta^\mu) + \mathcal{N}_t. \tag{3}$$

A critic is used to evaluate the policy estimated by the actor using the TD error:

$$y_i = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t). \tag{4}$$

As in the DQN method, experience replay is used to avoid correlations in the observation sequence.

Updating directly the actor and critic network weights with the gradients from the TD error signal causes instability, hindering convergence and learning itself. Target networks are

then used to generate the targets for the TD error calculation, which regularizes the learning algorithm and increases the stability of the solution, similarly to DQN. The TD target and critic's loss function are the following:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}), \tag{5}$$

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q)^2). \tag{6}$$

In the last equation, it is used a minibatch of size N sampled from the replay buffer, i being the sample index. The target for the TD error, yi, is calculated from the sum of the immediate reward and the outputs of the actor-target and critic-target networks, having weights $\theta^{\mu'}$ and $\theta^{Q'}$ respectively. Then, the critical output can be computed as $Q(s_i, a_i|\theta^Q)$

The interest of the DDPG algorithm for this work lies in the fact that it can be used to construct a policy of continuous decision, which can be used for the construction of a law of control of propulsion and rudder.

3 Simulation

The simulations made in this work have used two different simulators in order to train the RL agent to accomplish the navigation task.

The first one is the Simple Navigation Simulator (SNS), developed in the scope of this work, using a 3DOF dynamics and 5-Order Runge-Kutta Integrator, built using the dynamics described in (TANNURI, 2002) and (FOSSEN, 2011). This simulator is based on OpenAI Gym (BROCKMAN ; et al., 2016) environment, it is now a open-source available simulator under the name ShipAI (PEREIRA; REJAILI, 2018).

The second simulator used is the commercial software developed by the TPN group. The TPN simulator is the biggest ship maneuver simulator in Brazil. It can be summarized as a 4-order Runge-Kutta integration system, which integrates a set of 6 differential equations that govern the dynamics of the ship in its 6 DOF.

In the scope of this work, the ship was simulated in channel conditions, in the absence of wind, waves and current, and with navigation and control forces defined only by propeller and rudder actions.

For both simulators, the input parameters are the rudder and propulsion control forces. They have the vector form of $A_V = [A_l, A_p]$, where A_l is the dimensionless rudder command and A_p the dimensionless propulsion command, such that $A_l \in [-1, 1]$ e $A_p \in [0, 1]$. These parameters have a direct proportional relation with the rudder angle and the propulsion, such that:

$$\begin{aligned} A_p \in [0,1] &\rightarrow T_p \in [T_p^{\min}, T_p^{\max}], \\ A_l \in [-1,1] &\rightarrow \phi_L \in [\phi_L^{\min}, \phi_L^{\max}], \end{aligned} \tag{7}$$

where T_p refers to the propulsion force and ϕ refers to the rudder angle.

The output of both simulators are the global positioning and velocity, described by $(X, Y, \theta, v_x, v_y, \dot{\theta})$, where X, and Y refers to the global axis-positioning, θ refers to the attack angle, $v_x, v_y, \dot{\theta}$ are respectively the velocity in X, Y and the angular velocity.

4 Proposed Solution

4.1 Learning Strategy

The SNS simulator was built due to the need for an appropriate fast simulator to create a first learning of the RL agent, then the TPN simulator was used to perform a transfer learning step. The transfer learning step was made using a Starting-Point Method (SPM) (TORREY; SHAVLIK, 2010), i.e. using the model learned with the SNS as the starting point for the TPN simulator training. This process was used for both DQN and DDPG agents.

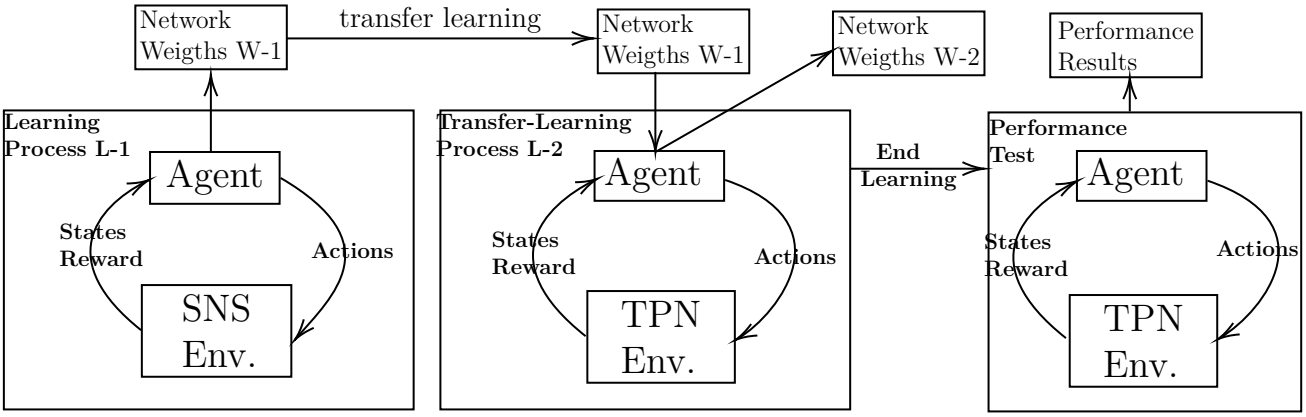

Figure 1 – Schematics of the learning procedure

4.2 RL structure

4.2.1 Navigation Task

The main purpose of the navigation task was defined as follows:

Use the rudder and propulsion controls to perform a defined linear navigation path along a channel.

The navigation trajectory was defined as the central mid-line of the navigation channel. The channel's configurations were inspired by the Suape channel, northeast of Brazil, which is straight, 5km long and 300m wide.

4.2.2 States

The states chosen for the application of RL in the task of the ship were as follows:

$$s = (d, \theta, v_x, v_y, \dot{\theta}), \tag{8}$$

where d is the distance from the center of mass of the ship to the guideline; θ is the angle between the longitudinal axis of the ship and the guideline; v_x is the horizontal speed of the ship in its center of mass (in the direction of the guideline); v_y is the vertical speed of the ship in its center of mass (perpendicular to the guideline); $\dot{\theta}$ is the angular velocity of the ship. Figure 2 shows these states in detail. Initial states were restricted to plausible values for ship entrance on a channel, and to yield convergence of the algorithm:

$$\begin{aligned} d_0 &\in [0, 30] \text{ m}, \\ \theta_0 &\in \left[-\frac{\pi}{15}, \frac{\pi}{15}\right] \text{ rad}, \\ v_{x0} &\in [1, 2] \text{ m/s}, \\ v_{y0} &\in [-0.4, 0.4] \text{ m/s}, \\ \dot{\theta}_0 &= 0 \text{ rad/s}. \end{aligned} \tag{9}$$

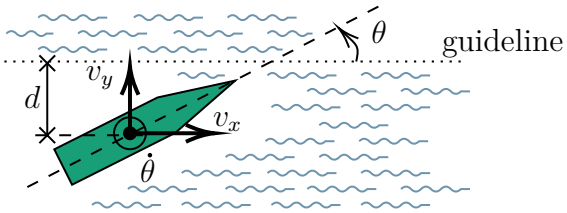

Figure 2 – Ship States

4.2.3 Control Actions

The control actions consist on the dimensionless rudder angle and propeller rotation, as described on the section 3. For the DQN, the actions were discretized on 21 rudder angles and 3 propeller rotations:

$$\begin{aligned} \text{SNS}: A_l^{\text{DQN}} &\in \{-1, -0.9, -0.8, ..., 0.9, 1\}, \\ \text{TPN}: A_l^{\text{DQN}} &\in \{-1, -0.9, -0.8, ..., 0.9, 1\}/3, \\ A_p^{\text{DQN}} &\in \{0, 0.1, 0.2\}. \end{aligned} \tag{10}$$

For the DDPG, the actions are continuous, but they were limited as follows:

$$\begin{aligned} A_l^{\text{DDPG}} &\in [-1/3, 1/3], \\ \text{SNS}: A_p^{\text{DDPG}} &\in [0, 0.2], \\ \text{TPN}: A_p^{\text{DDPG}} &\in [0, 0.24]. \end{aligned} \tag{11}$$

The heuristic of the control action limits was chosen after being proved sufficient to this paper's objective on preliminary tests on both SNS and TPN simulators. It also follows practical knowledge from actual pilots.

4.2.4 Reward Definition

The reward was designed to punish the ship's deviation from the guideline and the speed setpoint, with a small tolerance:

$$r(s) = k_{tol} - k_d \times \frac{|d|}{d_{\max}} - k_\theta \times \frac{|\theta|}{\theta_{\max}} - k_{v_x} \times \frac{|v_x - v_{sp}|}{v_{x_{\max}}} - k_{v_y} \times \frac{|v_y|}{v_{y_{\max}}} - k_{\dot{\theta}} \times \frac{|\dot{\theta}|}{\dot{\theta}_{\max}}, \tag{12}$$

where $k_{tol} = 1$ is the tolerance and defines the maximum deviation the agent can have still earning a positive reward; $\{k_d, k_\theta, k_{v_x}, k_{v_y}, k_{\dot{\theta}}\} = \{8, 8, 1, 1, 1\}$ are the weighted adjusts given for each setpoint deviation; $v_{sp} = 2$ is the speed setpoint; $\{d_{\max}, \theta_{\max}, v_{x_{\max}}, v_{y_{\max}}, \dot{\theta}_{\max}\} = \{150 \text{ m}, \pi/2 \text{ rad}, 2 \text{ m/s}, 4 \text{ m/s}, \pi/9 \text{ rad/s}\}$ are the maximum state values admitted.

4.3 Hyperparameters

The hyperparameters used are shown in the table below:

Table 1 – Methods settings

Methods	DQN with experience replay and target network	DDPG: Actor-critic, experience replay and target network
ANN's architecture	Feedforward, Input: states, 4 layers: [256,128,64,33] neurons, ReLU activation on hidden layers, linear on output, Output: action values, Optimization: Adam, Learning rate 1e−3	Actor: feedforward, Input: states, 3 layers: [400,300,2] neurons, ReLU activation on hidden layers, softsign on output, Output: action, Optimization: Adam, Learning rate 1e−4 Critic: feedforward, Input: actions, states, 3 layers: [400,300,1] neurons, ReLU activation on hidden layers, linear on output, Output: boolean (dimension = actions), Optimization: Adam, Learning rate 1e−3
Training	Steps: 400000, $\gamma = 0.99$ Policy: ε-greedy, ε linear annealed from 1 to 0.1, Experience replay memory: 20000 transitions, Hard Target Model Update: $C = 1000$	Steps: 600000, $\gamma = 0.99$, Stochastic process: Ornstein-Uhlenbeck ($\theta = 0.3$, $\mu = 0$, $\sigma = 0.3$), Experience replay memory: 20000 transitions Soft Target Model Update: $C = 1\text{e}{-2}$
Transfer Learning	SPM with 50000 steps, ε linear annealed from 0.1 to 0.01	SPM with 100000 steps, parameters equal to training

5 Results

To evaluate the performance of both methods, two test scenarios were proposed. The first one consists on 100 episodes with random starting states on the entrance of the channel, as shown on equation 9. The objective is to evaluate the survival of navigation, i.e. the absence of collision with the channel borders during episodes. Furthermore, one can verify generally the distance ($d \to 0$) and velocity($v_x \to 2$ m/s) convergence. For the second scenario, 10 episodes were ran in the TPN simulator with defined initial states - $s_i = [d_i, \theta_i, vx_i, vy_i, \dot{\theta}_i]$, where:

$$\begin{gathered} d_i = 30 \text{ m},\ \theta = \frac{k}{10} \cdot \frac{\pi}{15} \text{ rad } (k = 0, 1, 2, ..., 9), \\ va_i = 1.5 \text{ m/s},\ v_{xi} = va_i \cdot cos(\theta),\ v_{yi} = va_i \cdot sin(\theta),\ \dot{\theta}_i = 0. \end{gathered} \tag{13}$$

The objective of such experiment is to observe the states evolution for a gradually increasing initial angle, thus evaluating the controller performance. Convergence and rudder/propeller commands are then analyzed.

5.1 DQN

The DQN method results in a policy that is insufficient to control the speed and direction of the ship, presenting distance oscillations and velocities far from the setpoint. The first test scenario presented no collision, but the convergence to the distance is oscillatory with $d < 15$ m. The speed did not converge to the setpoint. The second scenario results are shown on figure 3.

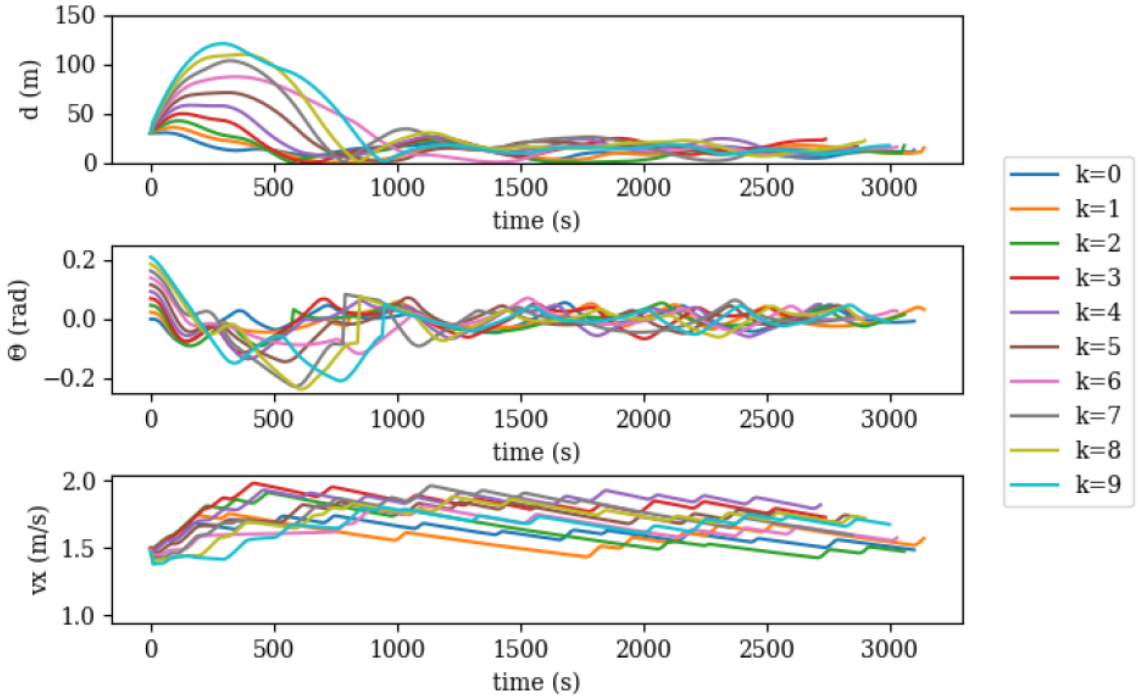

Figure 3 – States evolution with DQN after Transfer Learning, second scenario

5.2 DDPG

The DDPG method performed the task with good results, the ship followed the line with almost no oscillations, and the distance from the guideline tends to zero as we can see from the results below. The first scenario had no collisions and both distance and speed converged. The velocity reaches 90% of its setpoint around $t = 1380$ s for all cases. Figure 4 shows results for the second scenario.

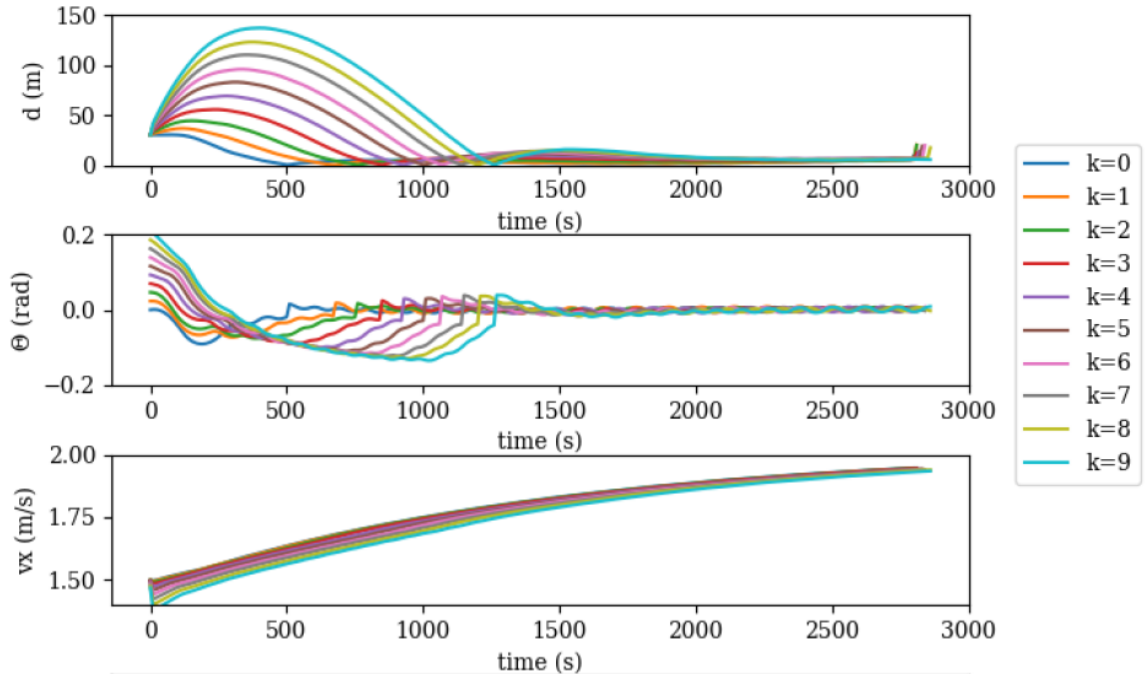

Figure 4 – States evolution with DDPG after Transfer Learning, second scenario

5.3 Control performance

One cannot evaluate the 10% settling time on DQN control, once it does not reach the 10% margin. However, for all scenario 2 tests, its rise time is around 1000 s. Comparing with DDPG, which has a rise time between 500 and 1200s, DQN has a smaller variance on this value. DDPG's best settling time for the distance is around 250 s ($k = 0$), while the worst is almost 1100 s ($k = 9$). The velocity settling time is around 1400 s for all k.

Still on scenario 2, both methods presented oscillating rudder actions, and while DQN also oscillated the thrust, DDPG maintained it almost constant.

6 Conclusion

In spite of fulfilling its objectives, the DDPG method presents some limitations for practical applications that have to be considered before real-life implementation, such as: the evaluation of system robustness response in situations not covered by the policy; the energy consumption analysis related to the resulting control law and the feasibility of implementation navigation given oscillating navigation patterns in the trajectory.

Some options can be explored to improve the control policy proposed in this paper such as: the inclusion of an incremental action policy for rudder and propulsion actions; the inclusion of a punitive factor for costly command actions (similar to LQR control techniques), the decoupling of control actions in two separate neural networks, a more profound analysis on the influence of the parameters on the controller performance (especially the reward weights and learning architecture/hyperparameters).

REFERENCES

AHMED, Yaseen Adnan; HASEGAWA, Kazuhiko. Experiment results for automatic ship berthing using artificial neural network based controller. *IFAC Proceedings Volumes*, Elsevier BV, v. 47, n. 3, 2014, 2658–2663. Disponível em: <https://doi.org/10.3182/20140824-6-za-1003.00538>.

AMENDOLA, José ; et al. Batch reinforcement learning of feasible trajectories in a ship maneuvering simulator. In: *Anais do XV Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*. [S.l.: s.n.], 2018.

BROCKMAN, Greg ; et al. Openai gym. *CoRR*, [S.l.: s.n.], abs/1606.01540, 2016.

CUTLER, Mark; HOW, Jonathan P. Efficient reinforcement learning for robots using informative simulated priors. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015. Disponível em: <https://doi.org/10.1109/icra.2015.7139550>.

DOUGHERTY, Mark. A review of neural networks applied to transport. *Transportation Research Part C: Emerging Technologies*, Elsevier BV, v. 3, n. 4, aug 1995, 247–260. Disponível em: <https://doi.org/10.1016/0968-090x(95)00009-8>.

FILHO, Asdrubal N. Queiroz; ZIMBRES, Marcelo; TANNURI, Eduardo A. Development and validation of a customizable DP system for a full bridge real time simulator. In: *Volume 1A: Offshore Technology*. ASME, 2014. Disponível em: <https://doi.org/10.1115/omae2014-23623>.

FOSSEN, Thor I. Handbook of marine craft hydrodynamics and motion control. John Wiley & Sons, Ltd, 2011. Disponível em: <https://doi.org/10.1002/9781119994138>.

GERLA, Mario ; et al. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In: *2014 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE, 2014. Disponível em: <https://doi.org/10.1109/wf-iot.2014.6803166>.

HAFNER, Roland; RIEDMILLER, Martin. Reinforcement learning in feedback control. *Machine learning*, [S.l.]: Springer, v. 84, n. 1-2, 2011, 137–169.

HETHERINGTON, Catherine; FLIN, Rhona; MEARNS, Kathryn. Safety in shipping: The human element. *Journal of Safety Research*, Elsevier BV, v. 37, n. 4, jan 2006, 401–411. Disponível em: <https://doi.org/10.1016/j.jsr.2006.04.007>.

KINGMA, Diederik P; BA, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, [S.l.: s.n.], 2014.

LACKI, Mirosław. Reinforcement learning in ship handling. *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, [S.l.: s.n.], v. 2, n. 2, 2008.

LILLICRAP, Timothy P ; et al. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, [S.l.: s.n.], 2015.

MNIH, Volodymyr ; et al. Human-level control through deep reinforcement learning. *Nature*, [S.l.]: Nature Publishing Group, v. 518, n. 7540, 2015, 529.

PEREIRA, Jonathas; REJAILI, Rodrigo Pereira Abou. *ShipAI*. [S.l.]: GitHub, 2018. <https://github.com/jmpf2018/ShipAI>.

RAK, Andrzej; GIERUSZ, Witold. Reinforcement learning in discrete and continuous domains applied to ship trajectory generation. *Polish Maritime Research*, Walter de Gruyter GmbH, v. 19, n. Special, oct 2012, 31–36. Disponível em: <https://doi.org/10.2478/v10012-012-0020-8>.

STAMENKOVICH, M. An application of artificial neural networks for autonomous ship navigation through a channel. In: *IEEE PLANS 92 Position Location and Navigation Symposium Record*. [S.l.: s.n.], 1992. p. 346–352.

TANNURI, Eduardo Aoun. *Desenvolvimento de metodologia de projeto de sistema de posicionamento dinâmico aplicado a operações em alto-mar*. Tese (Doutorado), 2002. Disponível em: <https://doi.org/10.11606/t.3.2002.tde-04082003-173204>.

TORREY, Lisa; SHAVLIK, Jude. Transfer learning. In: *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. [S.l.]: IGI Global, 2010. p. 242–264.

Jonathas Marcelo Pereira Figueiredo, graduating in Mechatronics Engineering at Escola Politécnica of the University of São Paulo, Brazil. Graduated in Automated Systems and Information Engineering at ENSE3 Grenoble-INP, France.

Rodrigo Pereira Abou Rejaili, graduating in Mechatronics Engineering at Escola Politécnica of the University of São Paulo, Brazil. Graduated in Robotics and Embedded Systems Engineering at ENSTA ParisTech, France, and Machine Learning and Data Science at University Paris Sud, France.