Proceedings of ASME 2019 38th International Conference on Ocean, Offshore and Arctic Engineering
OMAE2019
June 9-14, 2019, Glasgow, Scotland

OMAE2019-96120

PORT CHANNEL NAVIGATION SUBJECTED TO ENVIRONMENTAL CONDITIONS USING REINFORCEMENT LEARNING

José Amendola
University of Sao Paulo
Sao Paulo, Brazil
Email: jose.amendola@usp.br

Eduardo A.Tannuri
University of Sao Paulo
Sao Paulo, Brazil

Fabio G. Cozman
University of Sao Paulo
Sao Paulo, Brazil

Anna H. Reali Costa
University of Sao Paulo
Sao Paulo, Brazil

ABSTRACT

This paper proposes a machine learning agent for automatically navigating a vessel in a confined channel subject to environmental conditions. The agent is trained and tested using a Ship Maneuvering Simulator and is responsible for commanding the rudder, so as to keep the vessel inside the channel with minimum distance from the center line, and to reach the final part of the channel with a prescribed thruster rotation level. The algorithm is based on deep reinforcement learning method and uses an efficient state-space representation. The advantage of using reinforcement learning is that it does not require any expert to directly teach the agent how to behave under particular conditions. The novelty of this work is that: it does not require previous knowledge on the vessel dynamic model and the maneuvering scenario; it is robust against fluctuations of environmental forces such as wind and current; it considers discrete actions of rudder commands emulating the pilot actions in a real maneuver. The developed method is convenient for simulations in scenarios or areas that were never navigated before, in which no previous navigation data can be used to train a conventional supervised learning agent. One direct application for this work is the integration with a realistic fast-time maneuvering simulator for new ports or operations. Both training and validation experiments focused on the unsheltered approach channel of the Suape Port, in Brazil; these experiments were run in a SMH-USP maneuvering simulator (real environmental conditions measured on-site were employed in simulations).

1 INTRODUCTION

Navigation in restricted waters is a very complex topic and, even with autonomous ships about to be launched in a very near future, it still resists automation. Maneuvering in ports, bays and rivers depend on pilot experience regarding environmental conditions and navigational area particularities.

A very important application for ship maneuvering simulators are fast-time simulations for engineering purposes, where new navigation scenarios are tried out. One difficulty of fast-time simulations is the absence of previous recorded trajectories combining the same scenario and vessel configurations. Such simulations are usually performed by non-pilots and the trajectories are obtained by manually setting way-points and applying control algorithms used in automatic pilot systems. That might not represent real-world pilotage constraints properly and might not reveal the criticality of operations.

This paper employs a stable Deep Reinforcement Learning algorithm to solve the specific task of navigating through a straight and narrow channel. Reinforcement Learning (RL) is a field of machine learning based on the concept of reward. It does not require previous information of what the learning agent should exactly do at a given moment; it is thus highly recommended for problems where there are no information available, but there is a reasonable notion of how better certain actions or states are over others. RL can solve fast-time simulation issues by not requiring previous navigation samples and by naturally allowing discrete time and action constraints which are typical from human piloting.

Our learning agent acts on rudder level of an under actuated vessel initially positioned at the channel entrance and its goal it to reach the end of the channel without colliding with the lateral

Copyright © 2019 by ASME

buoys. It extends the scenario used in a previous paper [1] by considering environmental conditions (wind, waves and current) and using the state-of-art algorithm Deep Q Network.

The problem representation incorporates discrete levels of rudder angle actuation and discrete time steps between actions that closely match the action of a human pilot during a maneuvering. A compact and reusable representation of states was used to improve learning efficiency. It is later shown that the agent trained in a given scenario can perform the task with variations in environment parameters and the knowledge obtained can be transferred for accelerating learning for agents in other scenario settings.

Reinforcement Learning was applied to ship maneuvering in [2] for navigating until a goal in a squared geographic domain.

More recent applications using state-of-art deep architectures can be found in [3], where the task was to reach a channel entrance from random starting positions, and in [4], for path following task using Line-of-Sight strategy. In [1], as mentioned in Section 1, the channel topology and the task were the same as here in this paper, but no environment conditions were considered, the state space adopted was simpler and a batch RL algorithm called Fitted Q Iteration was used.

This paper is structured as follows: Section 2 introduces key concepts of Deep RL and the simulator used; Section 3 specifies the problem representation and scenarios and conditions used for the experiments; Section 4 describes the experiments run and analysis of results obtained; Finally, Section 5 shows our conclusions and future directions for the work developed.

2 PRELIMINARIES

In this section we introduce the simulator employed in the experiments and shortly review basic concepts about Reinforcement Learning and about the algorithm we adopted.

2.1 SMH Simulator

The TPN-USP Maneuvering Simulation Center is the largest Brazilian Ship Maneuvering Center, being composed by 6 simulators, being 3 classified as full-mission (immersive system with more than 270° of projection). The simulation can be executed in the real-time mode in one or several cabins simultaneously (single or multiplayer simulation) as indicated in Figure 1. The simulator is used for evaluation of new ports and operations, risk analysis, pilots and captains training. The same simulation software can also be executed in fast-time mode, in which an algorithm that represents the behavior of the pilot controls the ship and tugs. In this case, the maneuvers run in accelerated mode and the execution of a large number of runs is possible. This kind of maneuvering simulation is useful in the early stages of ports design, since the statistical analysis of the ship tracks can define the optimal layout of the maneuvering area or the limiting conditions.

FIGURE 1. TPN-USP Full maneuvering simulator

[5] and [6] present a more detailed description of the mathematical formulation and physical fundamentals of the SMH models.

The mathematical models represent the motion of a floating vessel at low speed in 6 DOF (degrees of freedom), subjected to the external forces due to the environmental and tugboats and to the control forces provided by the thrusters, propeller and rudder. The 6 DOF vessel dynamics differential equations are solved using explicit 4rd order Runge-Kutta integration method, considering the interaction with the fluid and the external forces acting on the hull. However, as a matter of simplicity, this section will only present the equations of motion for the horizontal plane. We adopt two different coordinate systems to derive the ship equations of motions, as shown in Figure 2. The system $OXYZ$ is earth-fixed (inertial system) and the system $Oxyz$ ship-fixed, with the origin on central point of the keel midship section. The center of gravity G is at the distance x_G ahead from the point o, ox is the longitudinal axis of the vessel directed to the bow, and oy is the transversal axis, pointing to port. The heading of the vessel ψ defines the angle between the ox and OX axes.

Following [7], the horizontal 3 DOF equations of motion referred to the body-fixed $Oxyz$ coordinate system, considering symmetry with respect to the axis ox, are given by:

$$(M+M_{11})\dot{u} - (M+M_{22})vr - (Mx_G+M_{26})r^2 = X_{ext}, \qquad (1)$$

Copyright © 2019 by ASME

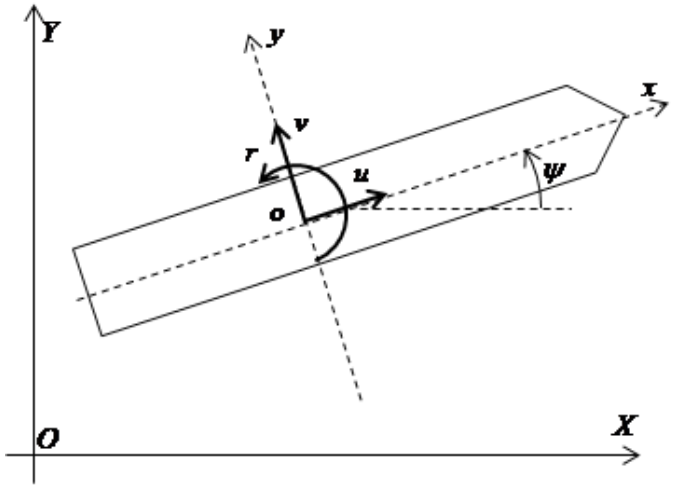

FIGURE 2. Ship coordinate system

$$(M+M_{22})\dot{v}+(Mx_G+M_{26})\dot{r}+(M+M_{11})ur=Y_{ext}, \quad (2)$$

$$(I_z+M_{66})\dot{r}+(Mx_G+M_{26})(\dot{v}+ur)+(M_{22}+M_{11})uv=N_{ext}, \quad (3)$$

where M is the vessel mass, I_z is the yaw moment of inertia of the ship, u and v are the surge and sway velocities respectively and r is the yaw angular velocity. The terms M_{11} and M_{22} are the ship added masses in the surge and sway directions, M_{66} is the ship added moment of inertia and M_{26} is coupled sway-yaw added inertia. The last term on the right side of the yaw equation is the Munk's moment. The subscript ext represents the external loads, that may be expressed in terms of different factors:

$$X_{ext}=X_h+X_w+X_{wv}+X_p+X_{tug}+X_M, \quad (4)$$

where X_h represents the hydrodynamic non-potential forces, including the current and maneuvering forces, X_w and X_{wv} represent the wind and wave forces, respectively, X_p represents the thrusters, propeller and rudder forces, X_{tug} represents the external action of the tug boats, either in contact with the hull or connected by a cable and X_M represents the forces due to mooring lines, fenders or anchor lines. In the present paper, the vessel will navigate along a channel without the help of any tugboats or attached to any lines, being only controlled by its own propeller and rudder under environmental forces. Therefore, the terms X_{tug} and X_M are null in the equation 4.

A physical-based model is used for the calculation of the rudder forces [8]. The rudder lift forces F_L are given by:

$$F_L(\beta_r)=0.5\rho A_r C_L(\beta_r)V_r^2, \quad (5)$$

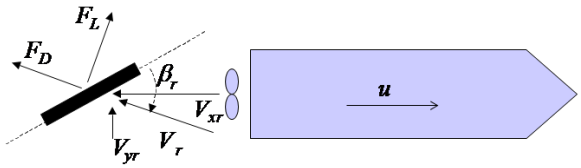

FIGURE 3. Forces on the rudder

where C_L is an dimensionless coefficient, β_r the effective rudder angle and V_r the relative velocity of the fluid onto the rudder.

Therefore, one must note that the rudder forces are dependent on the speed of the water on the rudder. This water jet is mainly generated by the propeller rotation, indicating that the vessel will be more controllable (maneuverable) if the propeller is operating. With engine stopped, the water flow on the rudder and the rudder forces will be reduced, making the vessel very difficult to be controlled.

2.2 Reinforcement Learning and Deep Q Network

Reinforcement Learning uses sequential decisions of an agent to learn how to behave [9]. At each time step, the agent observes a state, takes an action that results in a transition to another state and receives a reward signal. RL is based on the framework of Markov Decision Processes (MDPs); an MDP is described by a set of states S, a set of possible actions A, transition probabilities $p(s,a,s')$, $s,s' \in S, a \in A$, a reward function $S \times A \times S \to R$ and a discount factor $0 \leq \gamma \leq 1$. The latter factor attenuates future rewards and is applied in infinite horizon tasks to guarantee convergence of accumulated rewards. The lower the discount factor, the more short-sighted the agent. A Markov property is assumed by MDPs: every transition depends only on s and a.

RL algorithms look for optimal policies maximizing expected cumulative reward (a policy is a function from states to actions). The Q value is the expected accumulated reward if action a is taken at state s as specified by a policy π; we have that $Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} | s_t = s, a_t = a\right]$.

The Q-Learning algorithm is widely used in RL. The main idea of the algorithm is to update the Q-value of a given state-action pair during an exploration (online) by incrementing the Q value:

$$Q_t(s,a) \leftarrow (1-\alpha)\cdot Q_{t-1}(s,a)+\alpha\cdot(r+\gamma\cdot\max_{a'}Q_{t-1}(s',a')). \quad (6)$$

In the equation, α is the learning rate ($0 < \alpha \leq 1$). Updated this way, Q-learning can identify an optimal action-selection policy,

Copyright © 2019 by ASME

$\pi^*(s) = a$, for any given MDP, given infinite exploration time and a partly-random exploratory policy. Q-learning can be extended to continuous state-spaces, using for example supervised learning techniques to produce an estimate for the Q function.

Although the reward signal conceptually derives from interactions with the environment, from the point of view of engineering, it requires a human designer in order to define a reward function that is compatible with the problem representation.

Deep RL deals with algorithms combining RL with complex neural network architectures for approximating the action-value function. DQN, a special case of Deep RL, gained notoriety when it was used to solve ATARI games [10].

In DQN, two neural networks with same topology are used: at every iteration, the so-called primary network receives the state variables as input and outputs the respective action-values for each of the possible actions. The action with greater value is then selected. During the learning step, the secondary network, also called target network estimates the maximum target Q value for the subsequent transition. This value is then used to compute the error of the primary network. The weights of the target network are smoothly adjusted towards the values of the primary network weights. Using this secondary network, which slowly changes its weights, helps to reduce instability caused by the feedback of values used to compute the estimation.

Another source of instability are the correlations in the sequence of observations and changes in data distribution due to policy change. DQN tackles this problem with a technique called Experience Replay. It stores transition data in the replay buffer and selects a batch of transitions for fitting the network every learning step. The selection criteria can vary according to the problem an can even benefit experiences which do not occur very often.

Figure 4 illustrates the basic algorithm work flow: The actions generated by the policy are transmitted to the simulator and compose experiences along with reward, prior state and subsequent state. Such experience sets $\{ s,a,s',r\}$ are stored in replay memory for further mini-batch selection. The DQN block represents both neural networks: In the forward pass, the primary network outputs an action for state observed (policy); After the action is taken, occurs the backward pass, where the primary network is trained for the mini-batch and weight updating occurs for target network smoothly or with longer intervals using a gradient based algorithm.

There are several techniques for improving performance of RL based on Knowledge Transfer [11] that can also be applied to Deep RL [12].

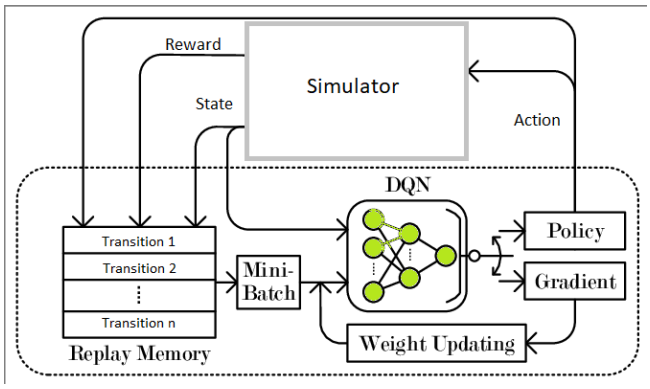

FIGURE 4. DQN ALGORITHM WORK FLOW.

3 METHODOLOGY

In this section we first present the scenarios considered in this work. We then show how the problem was formulated with RL, as well as which premises and simplifications were assumed.

3.1 Scenarios

An actual port channel located in the Suape Port, in the Northeastern Brazilian coast, was taken as geographic scenario for the experiments. The port channel is straight and is 210 m wide as shown in the nautical chart from Figure 5.

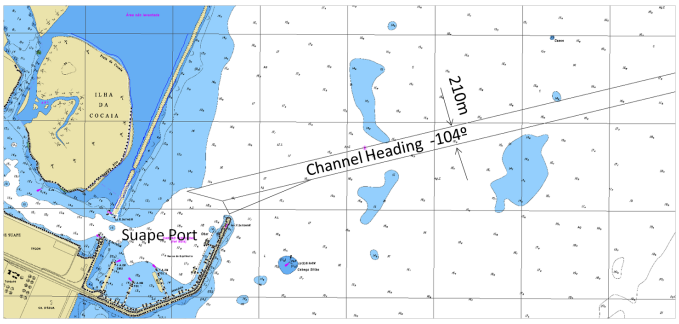

FIGURE 5. SUAPE PORT CHANNEL WITH BUOY DELIMITATIONS.

The model employed in the experiments is a tanker of class *Aframax* in loaded condition, whose main properties are listed in Table 1.

The typical environmental scenarios for that region sum up to 8 different conditions combining wind, current and wave. The conditions were grouped mainly according to direction of the wind

Copyright © 2019 by ASME

TABLE 1. MAIN PROPERTIES OF THE SHUTTLE TANKER.

Properties	Loaded Condition
Length Overall	244 m
Beam	42 m
Draft	15.3 m
Displacement	115.00 ton
Windage lateral area	2562 m^2
Main Engine Power	14280 kW

and current and the training process was applied to one condition of each group, but tested for all conditions inside the group, resulting in 3 agents. The grouping and the conditions considered are listed in Table 2 and illustrated in Figure 6. Agents were trained for conditions **1**, **3a** and **5**.

TABLE 2. ENVIRONMENTAL SCENARIOS

Group	Wind Speed (knots)	Current Speed (knots)	Wave Amplitude (m)	Wave Period (s)
A	Wind SE; Current N; Wave SE			
1	10.0	0.5	1.0	8.0
2	15.0	0.6	1.0	8.0
B	Wind NE; Current N; Wave SE			
3	20.0	0.4	1.0	8.0
4	25.0	0.8	1.0	8.0
3a	22.0	0.6	1.0	8.0
4a	23.0	0.7	1.0	8.0
C	Wind SE/S; Current N; Wave SE			
5	20.0 (SE)	0.8	1.7	10.0
6	20.0 (S)	0.8	1.7	10.0

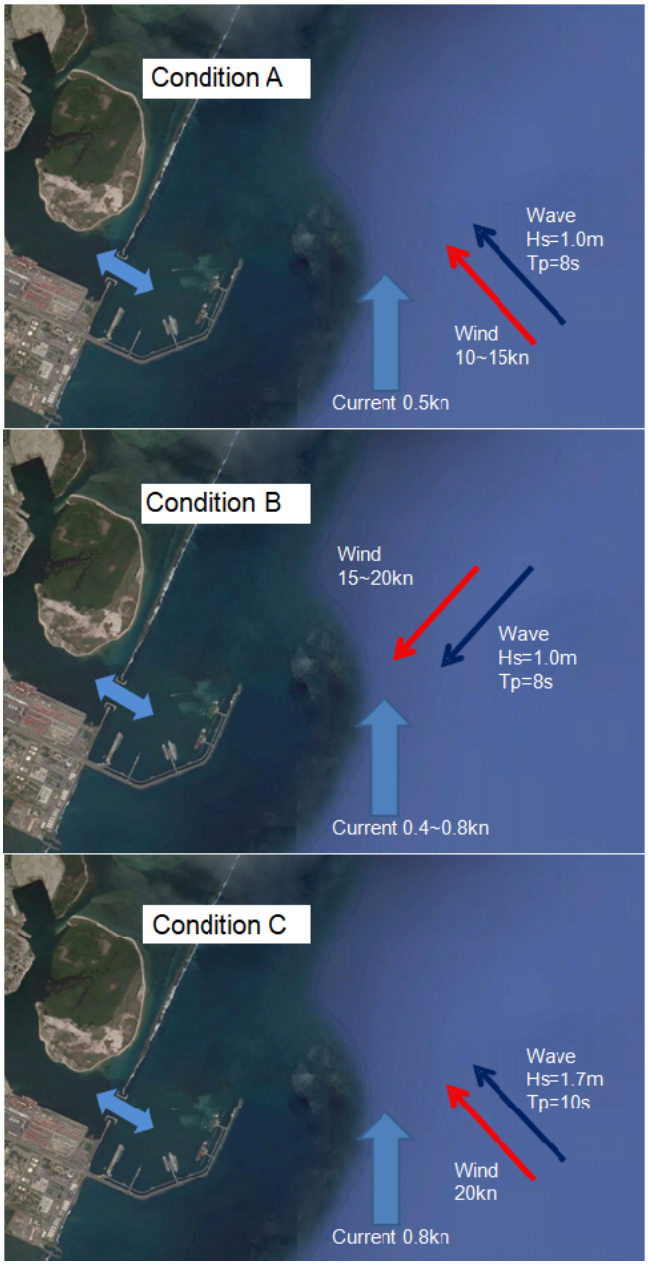

FIGURE 6. SCHEME OF ENVIRONMENTAL CONDITIONS ACTING ON THE CHANNEL AREA.

3.2 Problem Modeling as MDP

3.2.1 State representation

The complete vessel dynamics involves many state variables. However, most of these can be ignored without hampering task performance.

Although the goal is to reach the end of the channel, using information such as proximity to the goal or geographic position makes state space more complex and the agent would need to go exhaustively through episodes to experience transitions closer to the end. The problem was instead represented as an infinite horizon control problem. In this case, the agent has the goal of continuously keeping the vessel close to the channel center and to avoid collision. By accomplishing it, the vessel will eventually

Copyright © 2019 by ASME

reach the end of the channel.

The variables used are listed in Table 3. The variable *normalized lateral deviation* expresses how far from the center line of the channel and how close the vessel is from channel margins defined by buoys:

$$norm\_lat\_deviation = \frac{distance_{Starboard} - distance_{Portside}}{channelwidth}. \quad (7)$$

As represented by Equation (7), it is normalized by the channel width at the vessel position so that the agent trained stays robust against width changes along the channel: Figure 7 illustrates how variables are defined.

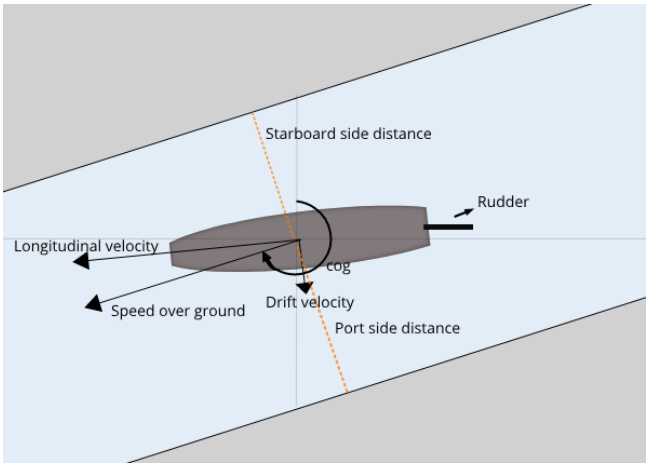

FIGURE 7. SCHEME ILLUSTRATING HOW VARIABLES ARE DEFINED.

TABLE 3. STATE VARIABLES.

State variable	Unit	Range
norm_lateral_dev	adimensional	-1.0 to 1.0
cog	degrees	243 to 270
rate-of-turn	degrees/s	-1.0 to 1.0
last_rudder_level	adimensional	$\{-0.5, -0.2, 0, 0.2, 0.5\}$

Course over ground is the direction of the actual speed of the vessel (measured at the mid-ship center point), even if the vessel moves misaligned with such line. The cog was considered, rather than orientation since the vessel navigates with drift angle to compensate environmental forces. Rate-of-turn represents the angular velocity which is positive for clockwise rotation and is essential due to the high rotational inertia.

The last commanded rudder level represents the last command given. Although its choice as relevant variable might not be intuitive, it works as a proxy indicator for vessel angular acceleration. When combined with rate-of-turn, it also reveals the effectiveness of the last command given, which might be crucial when dealing with external momentum due to environmental forces.

The range covered by the variables represents the limits established by the domain, with the exception of *cog*, which was delimited based on preliminary experiments. It was noticed that, although the vessel can present *cog* values beyond the range, it can hardly avoid collisions in cases where its course is highly displaced from the channel orientation. The range was then heuristically established with equal margins considering the channel orientation at the middle in order to accelerate the learning process and the policy convergence.

3.2.2 Actions

The action values represent the rudder command and are normalized by the maximum rudder angle (35^o). For the experiments, the rudder levels used were: -50%, -20%, 0%, 20%, 50%. The time interval chosen for transitions is $T_{transition} = 10s$, which means the vessel remains with same commanded rudder level for that amount of time.

3.2.3 Reward function

Simple reward functions can lead to a very slow learning convergence. Hence, it is recommended to project reward functions that encode a sense of proximity from the agent to its goal. In order to make learning process more efficient, an elaborated reward function which encodes information on how acceptable a transition is, must be employed. The agent must be encouraged to follow through the channel avoiding collision. Although most domains use slightly negative values for non-terminal regular states to avoid unexpected behavior such as in [13], the lack of good experiences that lead the agent to the goal can cause anticipated collisions to minimize overall negative punishment, as observed in [1]. In our problem, most transitions toward the goal are positive and partial knowledge on the domain is used to define higher values for states which are more acceptable and tend to lead to more successful missions. The reward function used is:

$$R(s,a,s') = \frac{1 - rot(cog - \theta_{channel})}{(1 + |norm\_lat\_deviation|)(1 + |cog - \theta_{channel}|)}. \quad (8)$$

The decay of the reward value should be proportional to the misalignment of the ship, for example. However, the same misalignment value, combined with rate-of-turn value which denotes the

Copyright © 2019 by ASME

tendency to further realignment, should result in a higher reward than a ship tending to follow straight in that direction or even diverge more from channel orientation. Such premises are compounded as multiplication or division operations. The numerator term will be higher than 1 if the ship tends to realign with the channel and lower than 1 otherwise. The left denominator term makes sure the reward is larger the closer the vessel is to the center; The right one causes the highest reward value to occur when the vessel navigates totally aligned with the channel. In the equation, the channel orientation is $\theta_{channel} = 256^o$.

Additionally, if the vessel collides at a given transition, the episode is reset and a punishment of -10 is added to the reward.

3.3 Exploration

The exploration policy used during trained was a variant of the epsilon greedy policy [9]. In this traditional policy, at a given training iteration, the agent has probability defined by the parameter ε of taking a random action rather than the action with maximum Q value. In the Annealed version of the policy used, the ε parameter is randomly chosen and can be as high as another parameter defined as ε_{max}. The parameter ε_{max} linearly decreases until it reaches a minimum value defined. For the experiments, ε_{max} was set to 0.8 initially and stabilizes at 0.2 after 3.10^5 iterations.

3.4 Neural Network

The neural network adopted is a regular Multilayer Perceptron [14] with input of dimension 4 (number of state variables) and output of dimension 5 (number of actions). The number of hidden layers used was 2 and the choice was based on another work [15] whose control task solved by RL had a similar complexity. The number of perceptrons for hidden layers was chosen as 64 and 32 respectively. The adopted activation function was ReLu (Rectifier Linear Unit) [16] and ADAM optimization algorithm [17] was chosen given its popularity in recent works. In such algorithm the weights of the perceptrons are updated, among other factors, by the gradient of error weighted by a learning rate.

4 EXPERIMENTS

In this section we present the values for parameters adopted in the experiments and we show the results obtained.

4.1 Parameters Setting

In every episode the vessel started aligned with the channel midpoint and velocity 3 m/s and training iterations were executed with simulator numerical integration step $T = 2s$. Although the simulator can have integration precision up to $T = 0.1s$, the computational time required to accomplish a reasonable number of iterations might be prohibitive since each iteration time length is inversely proportional to integration time step.

The learning rate for neural network was set to $\eta = 0.001$ and weights of perceptrons were randomly initialized with uniform distribution limited by a range inversely proportional to the number of inputs and outputs of the perceptron; The discount factor used was set to $\gamma = 0.99$; The weights from primary to target network in DQN algorithm was updated in "soft" mode, which means that at every iteration the weights from target were adjusted towards the primary network values by a factor $\theta = 0.002$.

The number of training iterations for all agents were defined after prior experiment in condition 1. The trajectory could be performed until the end of channel with reasonable stability after 3.10^6 iterations. Figure 8 shows that the accumulated reward per episode oscillates and increases significantly with this amount of iterations.

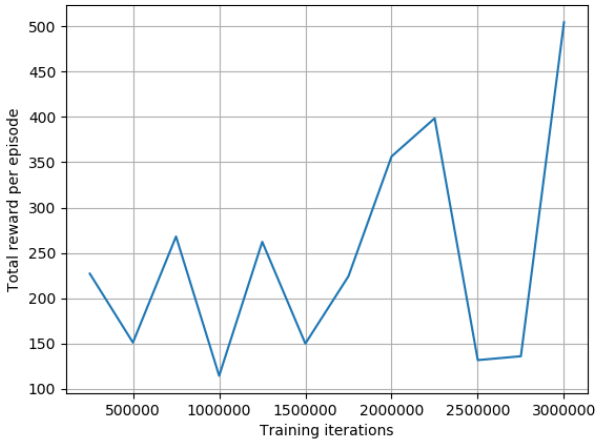

FIGURE 8. REWARD ACCUMULATED IN AN EPISODE FOR THE NUMBER OF TRAINING ITERATIONS.

4.2 Results

In order to evaluate the trajectories, the average normalized deviation was considered as well as its variance. Lower variances

Copyright © 2019 by ASME

indicate that the trajectory tends to be more straight and might contain less abrupt curvatures. The average vessel drift reveals the misalignment tendency between the vessel and the channel direction required to compensate environmental forces.

Tables 4, 5 and 6 show the values obtained for trajectories in groups A, B and C respectively. Figures 9, 10 and 11 illustrate such trajectories plotted over the channel.

TABLE 4. PERFORMANCE OF AGENT FROM GROUP A (TRAINED UNDER CONDITION 1).

Condition	Average Norm. Lateral Deviation	Norm. Lateral Deviation Variance	Average Drift (degrees)
1	−0.05	0.11	−2.4
2	−0.05	0.11	−3.2

TABLE 5. PERFORMANCE OF AGENT FROM GROUP B (TRAINED UNDER CONDITION 3A).

Condition	Average Norm. Lateral Deviation	Norm. Lateral Deviation Variance	Average Drift (degrees)
3a	0.26	0.19	−4.3
3	0.29	0.18	−3.7
4	0.35	0.09	−8.4 (failure)
4a	0.30	0.16	−5.2 (failure)

TABLE 6. PERFORMANCE OF AGENT FROM GROUP C (TRAINED UNDER CONDITION 5).

Condition	Average Norm. Lateral Deviation	Norm. Lateral Deviation Variance	Average Drift (degrees)
5	−0.34	0.06	−4.9
6	−0.34	0.06	−5.2

The training demonstrated that the agents trained under a certain condition can perform with very close behavior when magnitude of environment factors change within a certain range. The representation was also robust enough to keep commanding the vessel even at the end of the channel, where width gets larger. As in human pilotage, the RL agent successfully learned to compensate the environment forces by constantly navigating with a small drift angle.

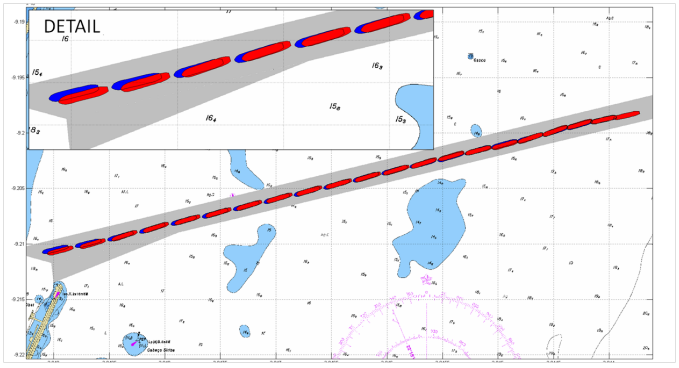

FIGURE 9. TRAJECTORY OF AGENT FOR GROUP A UNDER CONDITION SETTING 1 (BLUE) AND 2 (RED).

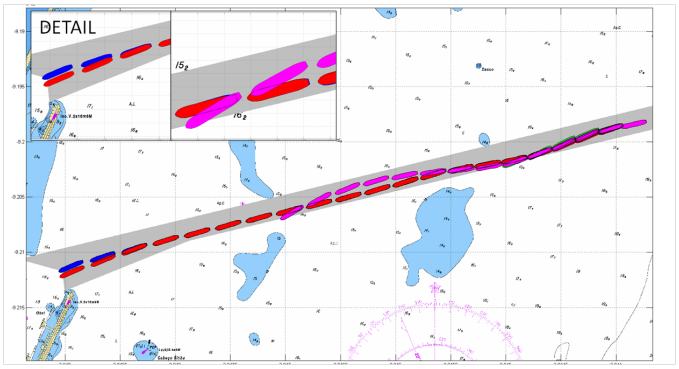

FIGURE 10. TRAJECTORY OF AGENT FOR GROUP B UNDER CONDITION SETTING 3A (BLUE), 3 (RED), 4 (GREEN) AND 4A (MAGENTA).

As expected, trajectories from group A presented lower average drift than C since the intensity of all environmental factors are stronger in C, making the navigation conditions more difficult. A vessel port side drift is also verified in the successful trajectories from group B (3 and 3a) since the vessel is in full draft and the current causes the largest forces on the vessel. The agent from group B was not able to navigate in every condition and collided when tested under conditions 4 and 4a. That happened because those conditions affected rudder efficiency and vessel drift in a way that the experiences during training could not extrapolate. Even the two successful trajectories from group B have shown that vessel navigated closer to the port side margin. That can be explained by the fact that wind and wave directions opposed current in this condition group and it induced the agent to have more positive experiences tending to port side during training.

Figure 12 shows the commanded rudder angle and the actual angle over time for navigation under Condition 1. The rudder orientation switches very frequently since the rate-of-turn must be maintained at low levels in order to stabilize the vessel. Although

Copyright © 2019 by ASME

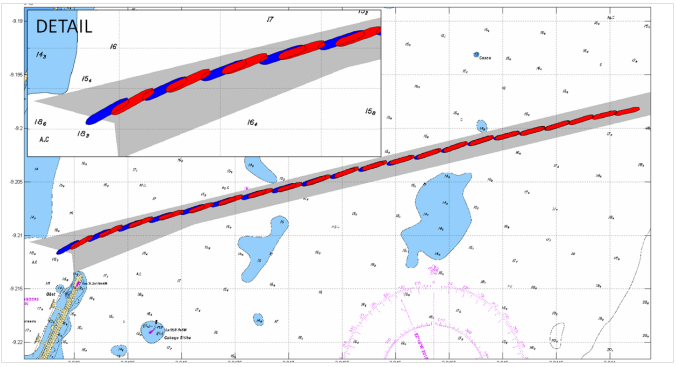

FIGURE 11. TRAJECTORY OF AGENT FOR GROUP C UNDER CONDITION SETTING 5 (BLUE) AND 6 (RED).

this behavior is acceptable and even occur in real-world piloting, the high variations in rudder level could be attenuated by inserting a small penalty for level variations in the reward function.

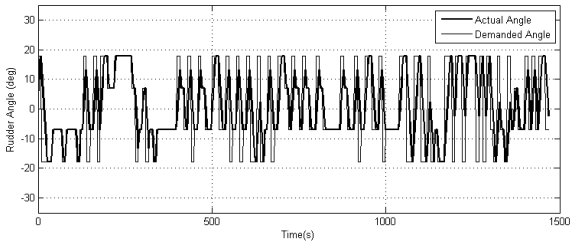

FIGURE 12. DEMANDED AND ACTUAL RUDDER ANGLES OVER TIME FOR CONDITION 1.

Since the agent trained under condition 3a was not successful in every other condition of group B, we extended the experiments by applying one of the possible techniques used for Knowledge Transfer. In this case, the weights of neural network trained in condition 3a were used to initialize the weights of the network of an agent to be trained under condition 4. After 9.10^5 (approximately 1/3 of the iterations used in the original training) iterations, the agent was able to successfully navigate until the end of the channel as shown in Figure 13.

Table 7 shows the results obtained for trajectory in condition 4 after Knowledge transfer. As expected, the average drift is larger than the trajectory obtained by the original agent from group B in conditions 3 and 3a since this condition has higher current intensity.

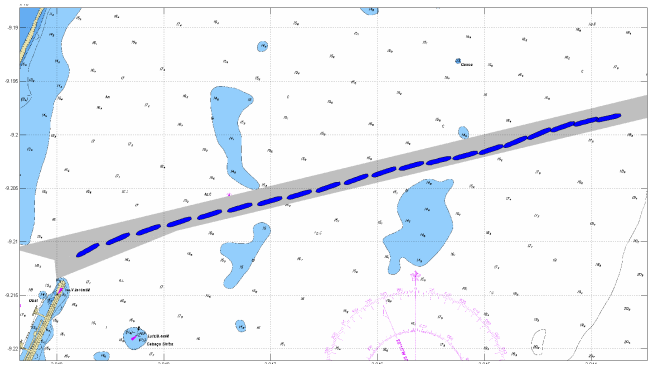

FIGURE 13. TRAJECTORY OF AGENT TRAINED FOR CONDITION 4 WITH WEIGHTS INITIALIZED FROM GROUP B AGENT.

TABLE 7. PERFORMANCE OF AGENT TRAINED FOR CONDITION 4 AFTER KNOWLEDGE TRANSFER.

Average Norm. Lateral Deviation	Norm. Lateral Deviation Variance	Average Drift (degrees)
0.14	0.12	−4.7

5 CONCLUSION AND FUTURE WORK

In this paper we have presented a Reinforcement Learning based solution for navigation in a port channel under realistic environment conditions. The algorithm enabled human-style maneuvering through discrete rudder levels and discrete interval between actions. The problem representation allowed the trained agent to navigate with variations on environment conditions and can be even extended to channels with different length and smooth width variations. It was also shown that knowledge transfer enabled an agent to be trained for a given scenario with reduced number of iterations.

Possible continuation of this work includes evaluating other state representations for handling more complex port geography such as curves and isles and also the extension of RL to the final phase of ship berthing, which includes commands to tugboats. Although this work focuses on the scope of fast-time simulations for project analysis, where obstacles are static, future works could also regard two-way channels with passing vessels underway.

ACKNOWLEDGMENT

Thanks to Petrobras for supporting the development of the maneuvering simulation center. The second author is partially supported by CNPq grant 304784/2017-6. The third author, by CNPq grant 308433/2014-9, and FAPESP grant 2016/18841-0. The fourth author, by CNPq grants 425860/2016-

Copyright © 2019 by ASME

7, 307027/2017-1 and FAPESP grant 2016/21047-3.

REFERENCES

[1] Amendola, J., Tannuri, E. A., Cozman, F. G., and Costa, A. H. R., 2018. "Batch reinforcement learning of feasible trajectories in a ship maneuvering simulator". *Anais do Encontro Nacional de Inteligłncia Artificial e Computacional (ENIAC)*, pp. 263–274.

[2] Lacki, M., 2008. "Reinforcement Learning in Ship Handling". *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation,* ***2***(2), pp. 157–160.

[3] Tuyen, L. P., Layek, A., Vien, N. A., and Chung, T., 2017. "Deep reinforcement learning algorithms for steering an underactuated ship". In 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 602–607.

[4] Martinsen, A. B., and Lekkas, A. M., 2018. "Straight-path following for underactuated marine vessels using deep reinforcement learning". *IFAC-PapersOnLine,* ***51***(29), pp. 329–334.

[5] Queiroz Filho, A. N., Zimbres, M., and Tannuri, E. A., 2014. "Development and Validation of a Customizable DP System for a Full Bridge Real Time Simulator". In International Conference on Ocean, Offshore and Arctic Engineering - OMAE 2014, Vol. 1A, p. V01AT01A047.

[6] Tannuri, E. A., Rateiro, F., Fucatu, C. H., Ferreira, M. D., Masetti, I. Q., and Nishimoto, K., 2014. "Modular Mathematical Model for a Low-Speed Maneuvering Simulator". In Proceedings of the 33th International Conference on Ocean, Offshore andArctic Engineering (OMAE2014), pp. 1–10.

[7] Fossen, T. I., 2011. *Handbook of Marine Craft Hydrodynamics and Motion Control*.

[8] Molland, A., and Turnock, S., 2007. *Marine Rudders and Control Surfaces*.

[9] Sutton, R. S., and Barto, A. G., 2018. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press.

[10] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D., 2015. "Human-level control through deep reinforcement learning". *,* ***518***, Feb., pp. 529–533.

[11] Lazaric, A., 2012. "Transfer in reinforcement learning: A framework and a survey". In *Adaptation, Learning, and Optimization*. Springer Berlin Heidelberg, pp. 143–173.

[12] Glatt, R., and Costa, A. H. R., 2017. "Improving deep reinforcement learning with knowledge transfer". In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA., pp. 5036–5037.

[13] Randløv, J., and Alstrøm, P., 1998. "Learning to Drive a Bicycle using Reinforcement Learning and Shaping". *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 463–471.

[14] Haykin, S., 1999. *Neural Networks: A Comprehensive Foundation*. International edition. Prentice Hall.

[15] Hafner, R., and Riedmiller, M., 2011. "Reinforcement learning in feedback control". *Machine Learning,* ***84***(1), Jul, pp. 137–169.

[16] Maas, A. L., Hannun, A. Y., and Ng, A. Y., 2013. "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In International conference on machine learning.

[17] Kingma, D. P., and Ba, J., 2014. "Adam: A Method for Stochastic Optimization". *CoRR,* ***abs/1412.6980***.

Copyright © 2019 by ASME