IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 31, 2020, accepted August 6, 2020, date of publication August 10, 2020, date of current version August 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3015661

Navigation in Restricted Channels Under Environmental Conditions: Fast-Time Simulation by Asynchronous Deep Reinforcement Learning

JOSÉ AMENDOLA[1], LUCAS S. MIURA[1], ANNA H. REALI COSTA[2], (Member, IEEE), FÁBIO G. COZMAN[3], AND EDUARDO AOUN TANNURI[3], (Member, IEEE)

[1]Numerical Offshore Tank Laboratory, University of São Paulo, São Paulo 05508-900, Brazil
[2]Intelligent Techniques Laboratory, University of São Paulo, São Paulo 05508-900, Brazil
[3]Department of Mechatronics Engineering and Mechanical Systems, University of São Paulo, São Paulo 05508-900, Brazil

Corresponding author: José Amendola (jose.amendola@usp.br)

The work of José Amendola was supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) under Grant 133230/2019-8. The work of Anna H. Reali Costa was supported by CNPq under Grant 307027/2017-1 and Grant 425860/2016-7. The work of Fábio G. Cozman was supported in part by CNPq under Grant 312180/2018-7, and in part by the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) under Grant 2019/07665-4. The work of Eduardo Aoun Tannuri was supported by CNPq under Grant 304784/2017-6.

ABSTRACT This paper proposes an efficient method, based on reinforcement learning, to be used as ship controller in fast-time simulators within restricted channels. The controller must operate the rudder in a realistic manner in both time and angle variation so as to approximate human piloting. The method is well suited to scenarios where no previous navigation data is available; it takes into account, during training, both the effect of environmental conditions and also curves in channels. We resort to an asynchronous distributed version of the reinforcement learning algorithm Deep Q Network (DQN), handling channel segments as separate episodes and including curvature information as context variables (thus moving away from most work in the literature). We tested our proposal in the channel of Porto Sudeste, in the southern Brazilian coast, with realistic environment scenarios where wind and current incidence varies along the channel. The method keeps a simple representation and can be applied to any port channel configuration that respects local technical regulations.

INDEX TERMS Fast-time maneuvering simulations, machine learning, deep reinforcement learning, ship path following control.

I. INTRODUCTION

Navigation in restricted waters is a very complex topic that resists automation, even as autonomous ships are planned for the near future. Maneuvering in ports, bays and rivers depends on the experience of pilots about the area and its environmental conditions. Moreover, navigation is affected by the complex hydrodynamic interactions of vessels with the margins and bottoms of channels.

In the context of maritime maneuvering simulators, fast-time simulations play a key Engineering role: They are an effective way of evaluating new navigation scenarios, thus identifying potential risks and characterizing the dynamic behavior of vessels. Such simulations are usually performed by non-pilots and do not demand a full-mission immersive simulator. Trajectories are obtained by setting way-points and applying control algorithms similar to those employed in automatic pilot systems [1]. The definition of useful way-points requires a trial and error procedure that is particularly difficult due to environmental conditions.

Another shortcoming of fast-time simulations is that they may produce maneuvers that do not match the ones selected by a human pilot. The trajectories obtained may either exceed the pilot's acceptance criteria regarding safety distances or may demand an unfeasible number of engine and rudder commands. Hence fast-time simulations cannot be used as a single tool to assess the quality of a maneuver; they must be combined with the judgment of experienced pilots.

These facts justify the use of machine learning techniques to build controllers within fast-time simulations so as to mimic human piloting. We are thus naturally led to reinforcement learning (RL), where an agent learns actions through experimentation. We resort to a distributed deep reinforcement learning algorithm to produce a fast-time

The associate editor coordinating the review of this manuscript and approving it for publication was Xiong Luo.

simulation environment that can navigate through restricted channels with realistic control actions and under the effect of environmental conditions.

Our controller acts on the rudder level of an underactuated vessel initially positioned at the channel entrance, aiming to reach the end of the channel without colliding with the limits defined by lateral buoys (we extend our previous work [2] and [3] on this topic). The commands are given in discrete levels and at discrete time intervals, as in human pilot navigation.

The novelty of this work lies in the following aspects:

- Planning algorithms for prior path generation are not needed; only on the channel limits and center line segments are used.
- Training in a single run can result in successful trajectories under a wide range of environmental conditions.
- Learning efficiency is improved by dividing a navigation scenario into a sequence of channel segments.

This paper is structured as follows. Section II introduces key concepts of Deep RL, related work and our simulator. Section III specifies the problem, its representation, and the algorithm we employed, while Section IV describes the experimental setup, the experiments and related analysis. Finally, Section V presents our conclusions and future directions.

II. PRELIMINARIES

In this section we present the fundamentals of reinforcement learning, a short review of related work, and the simulator we have used.

A. REINFORCEMENT LEARNING AND DEEP Q NETWORK

Reinforcement learning (RL) builds a policy for an agent out of actions and rewards [4]. At each time step during learning, the agent observes a state, takes an action that results in a transition to another state and receives a reward signal. This process is viewed as a Markov Decision Process (MDP), as it is assumed to have the Markov property, where a transition to a subsequent state s' depends only on the current state s and the action a performed on s.

An MDP is described by a *set of states* S, a *set of possible actions* A, a *transition model* $T(s, a, s')$, $T : S \times A \times S \rightarrow [0, 1]$ that yields the probability $p(s, a, s')$ of landing up in the new s' state given that the agent takes an action a in given state s, a *reward function* $R : S \times A \times S \rightarrow \mathbb{R}$ and a *discount factor* $0 \leq \gamma \leq 1$ that quantifies how much importance we assign to future rewards – the lower the discount factor, the more short-sighted the agent. In practice, a human designer must specify a reward function compatible with the problem and the task at hand.

The goal of a RL agent is to learn an actuation policy that allows it to collect as much reward as possible in its interactions. A *policy* $\pi : S \rightarrow A$ maps a state $s \in S$ observed by the agent to an action $a \in A$.

We define a state-action *value function* Q that estimates how desirable it is to perform action a in a given state s. The Q value,

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \mid s_t = s, a_t = a\right], \quad (1)$$

is the expected accumulated reward if action a is taken at state s as specified by a policy π. An *optimal policy* π^* is one that chooses the action with the highest value in $Q^{\pi^*}(s, \cdot)$ for each state s

$$\pi^*(s) = \arg\max_{a} Q^{\pi^*}(s, a). \quad (2)$$

We are interested in episodic tasks; that is, tasks that have different episodes, and an episode consists of all states between an initial state and a terminal state.

An RL agent interacts with its environment as follows: in each time step, the agent observes the state s and decides the action a to be taken and performs it; the environment then transitions to the next state s' and the agent receives a reward r as feedback. This cycle of interaction between the agent and the environment defines an *experience*, given by the tuple $\langle s, a, r, s' \rangle$. Experience are used to iteratively estimate the state-action value function Q.

Q-Learning is one of the most basic RL algorithms that uses a table to store the Q-values of all possible pairs of action and state [4]. The update rule of Q-Learning is

$$Q_t(s, a) \leftarrow (1 - \alpha) Q_{t-1}(s, a) + \alpha(r + \gamma \max_{a'} Q_{t-1}(s', a')), \quad (3)$$

where α is the learning rate ($0 < \alpha \leq 1$). The difference between the target value and the current value is commonly referred to as Temporal Difference (TD) Error. Q-learning can identify an optimal action-selection policy $\pi^*(s) = a$ for any 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.

Deep RL combines RL with complex neural network architectures so as to approximate the action-value function. Deep Q Network (DQN), a Deep RL algorithm, gained notoriety when it was used to solve ATARI games [5] and was adopted in our work because it is well-suited for problems where action space is discrete and state space is continuous.

One source of instability in training are the correlations in the sequence of observations and changes in data distribution due to policy change. DQN tackles this problem using a delayed copy of the neural network to estimate Q-values used in the TD error. It also adopts experience replay, where it stores transition data in a replay memory buffer and selects a batch of random transitions to fit the network at every learning step. The probability of choosing a transition can vary according to the problem and can even benefit from experiences that do not occur very often.

Another technique that improves DQN is Prioritized Experience Replay [6], where the sampling of experiences is biased proportionally to the TD error. The idea is to improve

sample efficiency by focusing learning on the most ''surprising'' experiences.

While Q-Learning and DQN are value-based algorithms and their goal is to infer Q-values, Policy Gradient (PG) algorithms act by optimizing estimators that directly output a continuous action or parameters for action probability distributions. They are applicable to a wider range of problems since they can produce stochastic policies and continuous actions. PG algorithms also have drawbacks: they have the tendency of converging to local optima and are less sample efficient.

In the present paper, Deep RL techniques were combined in a value-based approach that exploits the discrete action representation (Section III-C). Q-values estimated for a greedy policy can be intuitively interpreted in this domain. Q-values can also be useful in future applications for fast-time simulations in maritime domain, when inspired by ideas from robotics [7], [8].

The architecture we adopted extends DQN with Distributed Prioritized Experience Replay (APE-X); the resulting mix is known as APE-X-DQN. This latter architecture was proposed by Horgan *et al.* [9] and is described in Section III-E. For an analysis of training performance, APE-X-DQN is compared with the state-of-art PG algorithm Proximal Policy Optimization (PPO) [10] in Section IV-D. PPO has been rather popular within the PG category due to its performance.

B. RELATED WORK

Few efforts focused on RL within maritime simulators until some five years ago [11], [12]. The majority of recent papers combining Deep RL with ship control share similar state representations; they apply well known algorithms and differ mostly in terms of level of actuation and in the objective function. Most tasks can be categorized as a path following/tracking, path planning or collision avoidance. Recent proposals do not necessarily address exclusively one of the tasks, but usually bring novelty and focus to one of them.

Path following in maritime domain is modeled by RL as a control task where episodes may have indefinite length. They can either end in a non-desirable state or have an infinite horizon with the agent pursuing the desired set-point. [13] used the DDPG algorithm to follow a straight line; simulations used a 2-DOF model considering only yaw and sway velocities. [14] also used DDPG, with a reward function based on a Gaussian curve given by the cross-track error and the action continuously controlled the rudder angle. [15] extended this work so as to follow a curved path. The agent was first trained for a straight path and then trained in curved paths; the curves used for tracking were smooth and the derivative of path relative course was included as state variable. Simulations were performed for three vessel models with a 4-DOF model (including roll angle). Reference [16] used the same infrastructure employed in the current paper to apply state-of-art Deep RL algorithms to a path following control task without the need of line of sight strategy.

Path planning applications train the agent so as to provide desired positions from the starting point to the goal, taking the ship to be a particle; the generated path is then tracked using any control strategy [17]. Given a sea area with current, [18] used Q-learning in its tabular form by discretizing the sea area uniformly and using abstract discrete actions. That idea was extended to environmental conditions by [19], and the approach was similarly implemented by [20], [21]. Differently from path following applications, most path planning tasks are defined to achieve a goal for pre-established scenarios in a finite episode, considering distance to the goal as a state variable.

Although here we do not deal with dynamic obstacles, such as target ships, it is appropriate to review collision avoidance solutions. The literature focuses on autonomous ships instead of fast-time simulations analysis. Their state space representations include distance to the goal; reward functions include a factor proportional to goal proximity. [22] used a Boolean variable to indicate whether a dynamic obstacle is closer than a given range to the controlled ship and concatenates state variables from the last observations into a matrix; the latter was sent to a convolutional neural network to produce a compact representation. Actions were defined as increments in both yaw momentum and propulsion thrust, requiring an allocation technique in order to control rudder and propeller. The reward function consisted of a sum of factors such as constant penalty for each obstacle closer than delimited safe distance. Reference [23] not only dealt with path following conventional representation for each agent, but also variables from target ships in each agent state space. The reward function took into account path following and also compliance to COLREGS when ships approach each other. Reference [24] employed the line of sight strategy for path following in the state space and defined distance variable in sectors around the ship within certain range. The reward function was composed of a path following term and a collision avoidance term that penalized the weighted sum of sensor measurements. All reviewed collision avoidance proposals addressed the problem of non-stationarity of the environment by randomly generating multiple dynamic scenarios for training.

In our work, we focus on channel navigation with indefinite horizon. We assume a path is provided as a sequence of straight segments and we pay attention to collision with the margins, moving away from conventional path following tasks. In a previous publication [2] we looked at a straight channel without environmental conditions, and we used the Fitted Q Iteration algorithm. That work was later improved [3]: although curves were still not accepted, measurements of wind, current and wave forces were processed.

C. THE TPN-USP SHIP MANEUVERING SIMULATOR

The TPN laboratory at University of São Paulo maintains the largest Brazilian Ship Maneuvering Simulation Center. Its 4 full-mission simulators, 3 tug stations and 1 crane simulator can run independently or in integrated manner;

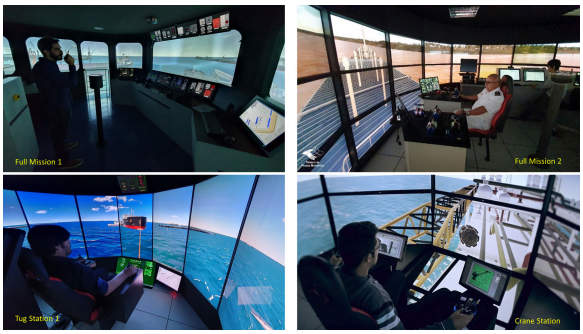

FIGURE 1. TPN-USP Ship Maneuvering Simulation Center.

they can represent different navigation or offshore scenarios (Figure 1). The Simulation Center is employed for evaluation of new ports and operations, risk analysis, and training of pilots and captains. The same simulation software can also be executed in fast-time mode, where an algorithm codifying the behavior of the pilot controls the ship and tugs – maneuvers then run in an accelerated mode, enabling the execution of a large number of runs under different conditions. Such a maneuvering simulation is useful in the early stages of ports design as the statistical analysis of the ship trajectories suggests layouts for the maneuvering area or their limiting conditions.

The mathematical model adopted in the TPN-USP simulators takes the motion of a floating vessel at low speed in 6 DOF (degrees of freedom), subjected to external forces due to the environment and tugboats, and to control forces provided by thrusters, propeller and rudder [25], [26]. The 6 DOF vessel dynamics differential equations are solved using a fourth order Runge-Kutta integration method, including interactions with the fluid and the external forces acting on the hull. For the sake of simplicity, this section only presents the equations of motion for the 3 horizontal DOF.

The Earth Fixed Global Reference Frame (GRF) coordinate system is represented by $oxyz$. The Local Reference Frame (LRF), $o_L x_L y_L z_L$, is attached to the vessel and centered in its midship position; it moves and rotates around z_L axis with the body. Figure 2 displays the geometry of the reference frames in the plane. The vector $\boldsymbol{\eta} = [x \;\; y \;\; \psi]^T$ gives the location of a vessel midship and its attitude in the GRF. In this representation, x is the easting, y is the northing and ψ is the yaw angle.

The floating rigid body dynamics yields the complete low-frequency equation of motion, including hydrodynamic inertial effects and external forces, as follows:

$$\begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_G - Y_{\dot{r}} \\ 0 & mx_G - Y_{\dot{r}} & I_z - N_{\dot{r}} \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} 0 & -mr & -mx_G r + Y_{\dot{v}} v + Y_{\dot{r}} r \\ mr & 0 & -X_{\dot{u}} u \\ mx_G r - Y_{\dot{v}} v - Y_{\dot{r}} r & X_{\dot{u}} u & 0 \end{bmatrix}$$

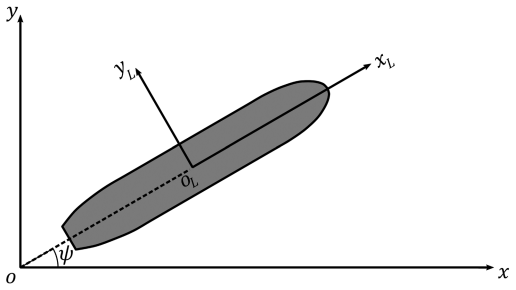

FIGURE 2. Reference Frames.

$$\times \begin{bmatrix} u \\ v \\ r \end{bmatrix} + (X_{\dot{u}} - Y_{\dot{v}}) \begin{bmatrix} 0 & r & 0 \\ r & 0 & 0 \\ -v & u_c - u & 0 \end{bmatrix} \begin{bmatrix} u_c \\ v_c \\ 0 \end{bmatrix} = \boldsymbol{F_{rudder}} + \boldsymbol{F_{prop}} + \boldsymbol{F_{tugs}} + \boldsymbol{F_{curr}} + \boldsymbol{F_{wind}} + \boldsymbol{F_{wave}}. \quad (4)$$

The vectors $\boldsymbol{\nu} = [u \;\; v \;\; r]^T$ and $\dot{\boldsymbol{\nu}} = [\dot{u} \;\; \dot{v} \;\; \dot{r}]^T$ are the midship vessel velocity and acceleration in the LRF respectively. Parameter m is the vessel rigid body mass, I_z is the moment of inertia around z_L, and x_G is the location of the center of mass with respect to the vessel midship position described in the LRF. The hydrodynamic derivatives $X_{\dot{u}}$, $Y_{\dot{v}}$, $Y_{\dot{r}}$, and $N_{\dot{r}}$ are the terms of the added mass matrix

$$\boldsymbol{M_A} = -\begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}} & Y_{\dot{r}} \\ 0 & Y_{\dot{r}} & N_{\dot{r}} \end{bmatrix}. \quad (5)$$

This model assumes an ocean current velocity of slow and irrotational variation that can be decomposed in the LRF as $\boldsymbol{\nu_c} = [u_c \;\; v_c \;\; 0]^T$. Symbols $\boldsymbol{F_{rudder}}$, $\boldsymbol{F_{prop}}$ and $\boldsymbol{F_{tug}}$ denote the vectors of rudder, propeller and tug induced forces, while $\boldsymbol{F_{curr}}$, $\boldsymbol{F_{wind}}$ and $\boldsymbol{F_{wave}}$ denote the vectors of environmental forces due to oceanic current, wind and first and second-order wave interactions.

III. PROPOSAL

In this section we present the formulation of our problem according to a RL framework. We detail the description of states, actions, state transitions, reward function, and discount factor. Finally, we describe the algorithm that finds the desired policy.

A. STATE VARIABLES

Table 1 lists all states variables of interest, and Figure 3 depicts them. In our setting, the propeller command remains fixed, so the agent does not control velocity [3]. This reduces

TABLE 1. State variables.

State variable	Unit	Range
dist_to_path	m	-100 to 100
cog_error	rad	$-\pi$ to π
rate_of_turn	rad/s	-1.0 to 1.0
lat_wind_vel	m/s	-8.0 to 8.0
lat_curr_vel	m/s	-1.0 to 1.0
dist_to_waypoint	m/s	0 to 1000

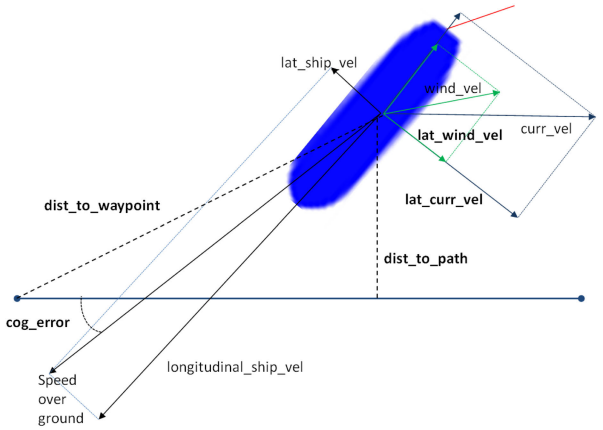

FIGURE 3. State variables (in bold).

the dimensionality of the problem, as all variables related to the vessel velocity and longitudinal effects of wind and current are discarded. Unlike previous work, no information regarding the margins was directly used in the state representation, so as to make the solution more robust against variations in margin geometry.

The state variable *dist_to_path* is the distance from the vessel's center point to the desired path segment. The value is negative when the vessel is at the left side of the path pointing towards the channel end. The state variable *cog_error* denotes the difference between the orientation of the path segment and the *course over ground (cog)*. The *cog* was considered, rather than heading angle, as the vessel can navigate with drift angle to compensate for environmental forces. The variable *rate_of_turn* denotes the angular velocity, which is positive for clockwise rotation and is important due to the high rotational inertia. The variables *lat_wind_vel* and *lat_curr_vel* denote the components of wind and water current velocities acting perpendicular to the ship. These variables help the policy to compensate for environmental forces that push the vessel towards the margins. Finally, state variable *dist_to_waypoint* carries information on how far the vessel is from the next way-point and is important to anticipate the actions required for changing the *cog*.

As suggested by Bhatt *et at.* [27], normalization of state variables so that they vary in a similar range can enhance learning performance. The variables were normalized between their extreme values observed in prior experiments. The dividing factors for variables can be found in Table 2.

TABLE 2. Dividing Factors.

State variable	Dividing Factor
dist_to_track	100
cog_error	0.15π
rate_of_turn	0.0015π
lat_wind_vel	4
lat_curr_vel	0.2
dist_to_waypoint	1000

B. TASK SEGMENTATION AND CONTEXT VARIABLE

Our initial experiments with curved narrow channels indicated a difficulty with the states described in the previous section. An experiment was executed in a channel with two 90° curves to opposite directions separated by a straight segment. The agent was to track the centerline segments generated by the channel margin. The agent learned to navigate in the first curve and to keep track of the straight middle segment, but failed in the second curve. That happened because the state-space representation did not include information about the ship position relative to the extent of the channel nor information about the curvature. As the RL agent could not distinguish any difference between these two states, it could not learn the correct policy. If the policy were to simply track the closest segment, it would eventually start to track another highly misaligned segment, causing collision with channel margins. Analogously, if the policy is learned with the second curved segment, it may fail the first one. This problem is illustrated in Figure 4 (top). The problem could be solved by tying solutions to specifics of each part of the channel. However, the policy for certain channel parts would no longer be applicable to other similar parts (e.g. straight parts of a channel would not no longer be handled by same representation due to different positions along the channel). Figure 4 (bottom) illustrates this issue.

The solution found to achieve a robust policy was, first, to segment the navigation in different RL episodes, and, second, to parameterize the Q function by a context variable.

Episode Segmentation: Each RL episode is delimited by three way-points in the channel centerline with a fixed distance $L_{segment}$ between them. That creates two desired path segments that the vessel must track, as seen in Figure 5. Instead of channel center lines, those segments are used as reference in order to standardize episode length span and to improve learning efficiency. The distance $L_{segment}$ cannot be too short so as to avoid trajectory instability in face of small angles, and it cannot be too long so as to avoid reducing the navigable area. The episode ends as soon as the vessel follows the two desired path segments and reaches the second way-point ahead. The segmentation improves sample efficiency, as experiences from a portion of channel can be used for similar situations in any channel position.

Context Variable: Pursuing two path segments with different angles under environmental forces demands a policy that succeds in a diversity of situations. The higher the angle between path segments, the more the agent must anticipate the rudder action before finishing the actual segment. We included context variables in the Q-value function to discriminate the context of the episode that the agent is facing. This RL technique was proposed by Schaul *et al.* [28]; it consists of extending the input of the value function with information about the context of the task being performed by the agent, so $Q(s, a)$ becomes $Q(s, c, a)$. This enables multi-task generalization as the policy can use information that discriminates which task the agent is facing. In our formulation, the context variable is the smallest difference between the orientation of

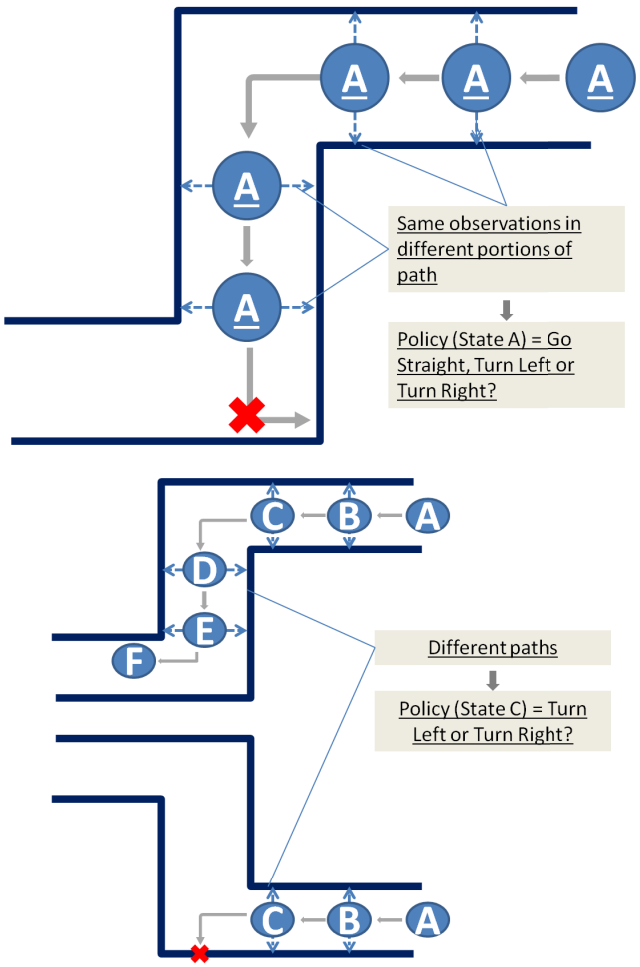

FIGURE 4. Top: a compact representation with no information about vessel's position may "confuse" the policy. Bottom: Vessel's position included in the state representation prevents the reuse of policy in different parts of the channel.

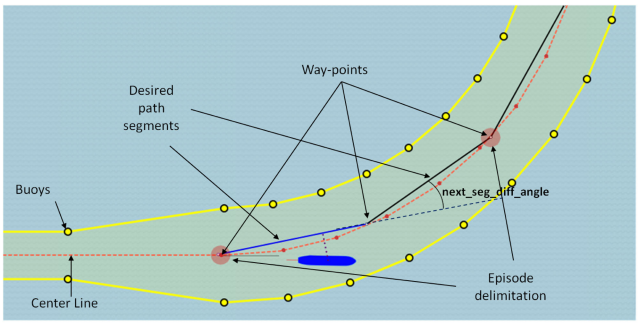

FIGURE 5. Channel partitions.

path segment ahead of the next way-point and the orientation of the desired path segment being followed, also referred as *next_seg_diff_angle* in Figure 5 (for segments at the end of the channel, this angle is set to zero).

C. ACTIONS

In order to better represent human piloting, rudder commands are given in discrete values and with minimum time interval between them. The action values represent the rudder command and are normalized by the maximum rudder angle (35°). The rudder levels used were: −50%, −20%, 0%, 20%, 50%. The propulsion level is fixed at 60% of maximum power along all trajectory. The time interval chosen for transitions is $T_{transition} = 10s$, which means the vessel remains with the same commanded rudder level for that amount of time.

D. REWARD

Randløv and Alstrøm [29] argued that negative values prevent the agent from going through states unnecessarily to accumulate positive rewards. Nonetheless, in our previous work [2], reward functions with negative values for most state-space led the agent to quickly go for collisions in order to minimize overall negative punishment. Because the limited available space in a channel prevents the vessel from navigating away from the centerline, positive rewards are not an issue.

Thus we selected:

$$R(s, a, s') = R_{cog}R_{dist}R_{rot} + R_{collision}, \tag{6}$$

ensuring that reward is negative only for collision states; factors are multiplied to provide a more steep decrease of reward as states become undesirable. The reward terms are defined as follows:

$$R_{cog} = \frac{1}{(1 + K|scaled\_cog\_displacement|)}, \tag{7}$$

$$R_{dist} = \frac{1}{(1 + K|scaled\_distance\_to\_track|)}, \tag{8}$$

and

$$R_{rot} = \frac{1}{(1 + K|scaled\_rot|)}, \tag{9}$$

as they behave in the same way as the vessel moves away from the track and starts rotating respectively. If the vessel collides, a −10 punishment value is added to the function:

$$R_{collision} = \begin{cases} -10 & \text{if } ship\_collided, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

To understand the effect of parameter K, preliminary experiments demonstrated that $K = 10$ improves the punitive behavior when vessel deviates from desired states; thus $K = 10$ was adopted. Figure 6 uses the example of *rate_of_turn* to show how the factor K affects the reward term. The reward term function is plotted with $K = 1$ and a more punitive factor $K = 10$.

E. LEARNING ARCHITECTURE AND ALGORITHM

We used the APE-X-DQN architecture for learning (Figure 7) [9]. It consists of two parts that share a Replay Memory Buffer: Actor and Learner. The actor part is actually composed of several actors responsible for interacting concurrently with separate environment copies that, in our case, are instances of the fast-time simulator. The role of each actor is to evaluate a policy and to store the experiences observed in a replay memory buffer; its policy is implemented as a

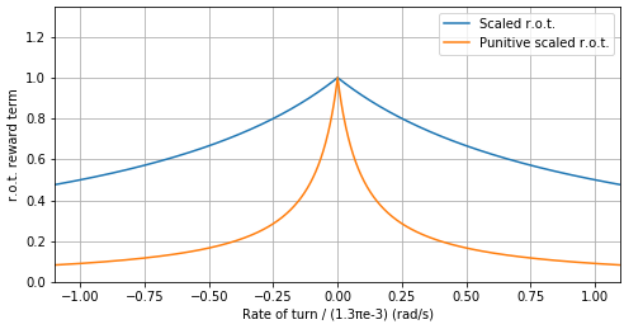

FIGURE 6. Effects of *rate_of_turn* punishment; $K = 1$ for Less Punitive Behavior; $K = 10$ for Punitive Behavior (adopted value).

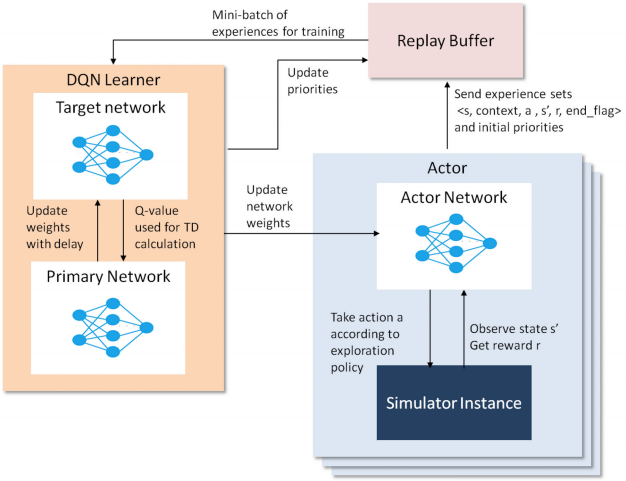

FIGURE 7. Architecture: APE-X-DQN.

deep neural network. The second part of the architecture consists of a learner that is responsible for sampling batches of data from the replay memory buffer to update the policy parameters. The learner periodically communicates updated network parameters to the actors. The replay memory buffer is centralized and shared between the actors – who feed it – and the learner – who consumes its data.

An actor is instantiated in each computing core available. The instance contains a copy of the policy neural network and its function is to collect experiences by interacting with an instance of the fast-time simulator. Actors explore different parts of channel under different current and wind conditions concurrently. The collected experiences consist of the state s where an action was performed, the action a itself, the state s' obtained after action, the reward signal r, the context variable c and the *end_flag*, indicating whether the state s' is final in the episode.

The experience set $< s, a, s', r, c, end\_flag >$ of an actor is sent to an actor local buffer. Batches with b experiences from the local buffer have their initial priority value calculated locally and then are sent to the replay memory buffer. Priority calculation is given by the TD error using Q-values estimated with local actor network parameters, as:

$$TD_{actor} = r_t + \gamma \max_a Q(s_t, c_t, a, \theta_{actor}) - Q(s_t, c_t, a, \theta_{actor}), \quad (11)$$

where in time step t, r_t is the reward, s_t is the state, c_t is the context, a_t is the action, γ is the discount factor and θ_{actor} represent the actor network parameters.

The actor's policy network parameters θ_{actor} are synchronized with the primary network parameters at every T_{actor} experiences collected from it. Each episode ends either if the vessel collides, reaches the end of the channel or finishes tracking two path segments. In order to balance the learning process throughout the channel, actors initialize the vessel at their simulator instance randomly in one of the defined way-points after the vessel collides or reaches the end of channel. The exploration policy adopted by the actors is the ϵ-greedy strategy that selects a random action with probability ϵ and acts greedily with probability $1-\epsilon$. The actors act while the training is carried out by the learner.

Algorithm 1 describes the interaction between each actor and the simulator instance.

Algorithm 1 Actor of APE-X-DQN

```
Initialize simulator instance with channel buoys, current map and wind map
Define waypoints at channel center line with distance of L_segment from each other
Initialize neural network with received parameters
local buffer ← ∅
repeat
    Observe state s and context variable c
    repeat
        a ← ε-greedy(s, c) and execute a
        Observe s'
        Get reward r
        Check if s' is final and set end_flag
        Add < s, c, a, s', r, end_flag > to local buffer
        if local buffer ≥ b then
            Compute priorities for experiences in local buffer (Eq. 11)
            Add b prioritized experiences to Replay Memory Buffer
        end if
        Every T_actor update θ_actor ← θ
        s ← s'
    until episode ends
until no more episodes
```

The learner in the APE-X-DQN architecture employs the DQN algorithm [5] and consists of a deep neural network that estimates Q-values for the policy. The learner periodically sends the actors the weights of this neural network so that they can update their respective neural networks. The training of the learner occurs in mini-batches B of experiences sampled from the Replay Memory Buffer D. A gradient-based

algorithm is applied to minimize the loss function with the following learning rule:

$$\mathcal{L}_t(\theta) = 1/2(G_t - Q(s_t, c_t, a_t, \theta))^2, \tag{12}$$

which is basically the squared TD error with

$$G_t = r_t + \gamma \max_a Q(s_{t+1}, c_{t+1}, a, \theta^-), \tag{13}$$

where θ represents the primary neural network parameters, and θ^- represents the target network parameters. The target network provides the Q-value for TD Error estimation. Its weights θ^- are adjusted to the primary network weights θ every time T_θ. Using a slow moving copy of the primary network in TD estimation improves learning stability.

Following the original DQN paper suggestion for further stability improvement, the loss function was taken as squared TD error (Equation 12) when the error value was between -1 and 1. The loss was replaced by the absolute TD error ($|G_t - Q(s_t, c_t, a_t, \theta)|$) outside this interval.

In order to prioritize training for experiences that are more "surprising" [9], the probability of an experience set being sampled from the replay memory buffer to a batch is modulated by its priority factor. The experience priority is updated by the learner primary network. The update is also performed using TD calculations, as in Equation 11, but employing the primary network parameters θ instead. Each training iteration to the primary network occurs at every $N_{steps-per-iter}$ experiences collected from actors. Algorithm 2 describes the training process in the learner.

Algorithm 2 Learner of APE-X-DQN

1: Initialize Replay Memory Buffer D
2: Initialize primary action-value function $Q(s, c, a, \theta)$ with random weights θ
3: Initialize target action-value function $\hat{Q}(s, c, a, \theta^-)$ with weights $\theta^- = \theta$
4: **repeat**
5: Sample a prioritized batch B of experiences from D
6: Compute loss function $\mathcal{L}$ (Eq. 12)
7: Update parameters θ using supervised learning and $\mathcal{L}$
8: Every T_θ steps reset $\theta^- \leftarrow \theta$
9: Compute priorities for D
10: Periodically remove old experiences from D
11: **until** no more learning

IV. EXPERIMENTS

In this section we describe experiments with a real scenario. We describe the area setting and vessel type in Section IV-A; then we describe the current and wind conditions in Section IV-B. The parameters adopted in the training are given in Section IV-C; we analyze the training process and policy performance in Section IV-D, and finally we evaluate the robustness of policy obtained in Section IV-E.

A. NAVIGATION AREA AND VESSEL TYPE

The proposed method was analyzed in navigation simulations in the Porto Sudeste Access Channel. It is located in Sepetiba Bay, State of Rio de Janeiro, in the Southeast region of Brazil (Figure 8).

FIGURE 8. Location of the Porto Sudeste Access Channel.

The simulated section of the access channel is approximately 16km (8.6 nautical miles) long, in three straight parts with a 58°, 85° and 30° heading, respectively. The minimum depth of the channel is 19m and the minimum width is 206m. There is a bend to access the final stretch of the channel, with a radius of 1600m (Figure 9).

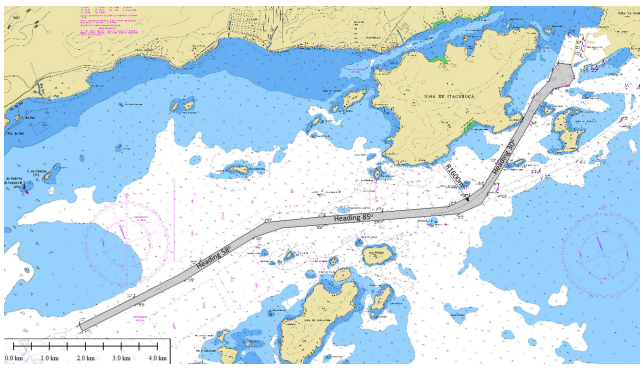

FIGURE 9. Porto Sudeste Access Channel over the Nautical Chart.

The vessel type used was a full loaded Suezmax DP tanker (160,000DWT). Although this ship has bow and stern thrusters, the navigation along the access channel was carried out only with the machine (propeller) and rudder. The ship's characteristics in shallow and deep waters are shown in the Figure 10.

B. CURRENT AND WIND CONDITIONS

The training and simulations were executed in different scenarios. Both Flood and Ebb current conditions were considered, with speed (measured at the main curve) from 0.4 to 1.0kn. The 20kn W and N wind were associated with the flood and ebb scenarios, respectively. Figure 11 shows the current in the navigation channel obtained from a validated hydrodynamic model. Table 3 shows the simulated environment.

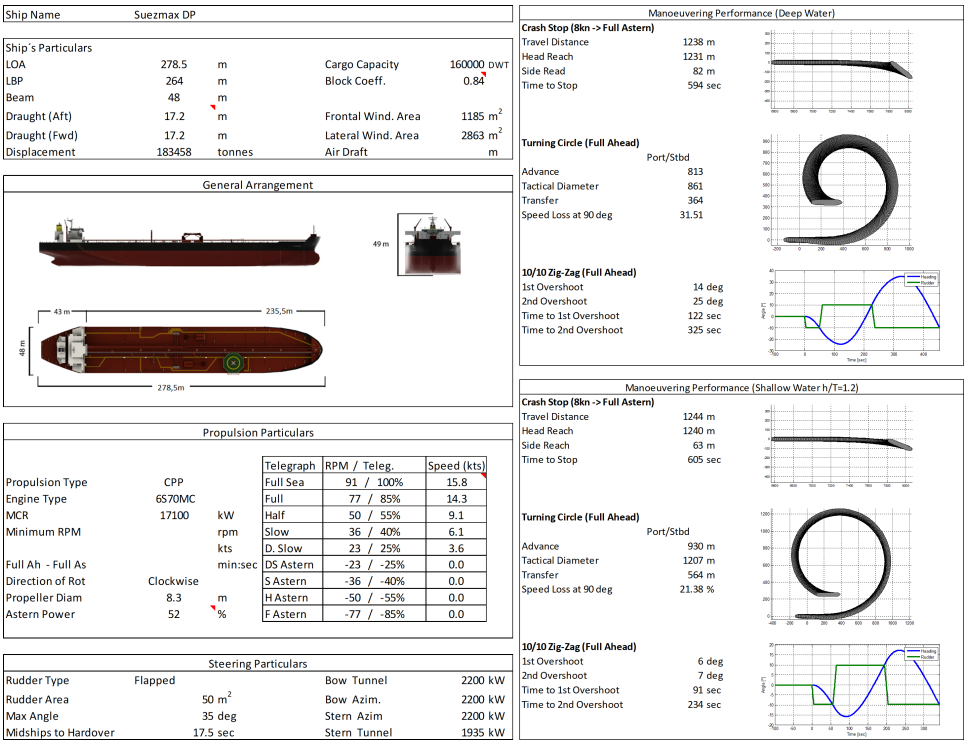

FIGURE 10. **Suezmax DP tanker: (left) main characteristics; (right) maneuvering properties.**

TABLE 3. **Environmental Scenarios (used for validation/training).**

Scenario	Current	Wind
E04(0.4kn) to E10(1.0kn)	Direction: Ebb Speed: 0.4kn; 0.5kn; 0.6kn; 0.7kn; 0.8kn; 0.9kn ; 1.0kn	Direction: N Speed: 20kn
F04(0.4kn) to F10(1.0kn)	Direction: Flood Speed: 0.4kn; 0.5kn; 0.6kn; 0.7kn; 0.8kn; 0.9kn ; 1.0kn	Direction: W Speed: 20kn

C. TRAINING SETTINGS

Because the agent learned navigation under different environmental conditions, massive experience collection was required, so the architecture was combined with DQN using 48 CPU cores with 2.5GHz of clock frequency and 128GB of RAM memory. The framework used for implementation was RLLIB [30] with the Python programming language.

The neural network adopted is a regular Multilayer Perceptron [31] with input of dimension 7 (number of state variables and a context variable) and output of dimension 5 (number of actions). The number of hidden layers used was 2 and the choice was based on a previous effort whose control task solved by RL had a similar complexity [32]. The number of perceptrons for hidden layers was chosen as 64 and 32, respectively. The adopted activation function was ReLu (Rectifier Linear Unit) [33] using ADAM optimizer [34]. The weights of the perceptrons are updated, among other factors, by the gradient of error weighted by a learning rate, defined as $l_r = 0.001$. The target network was updated with the primary network weights at a rate of $T_\theta = 8.10^5$ steps of experience tuples collected.

The replay buffer size was set to comprise at most $N_{buf} = 4.8.10^5$ steps. The number of experiences collected until a training iteration occurs was set to $N_{step-per-iter} = 1000$ and the batch size to $B = 128$. Exploration was set to decrease linearly from $\epsilon = 100\%$ to $\epsilon = 10\%$ in 2.10^6 steps, as seen in Figure 12 and each actor stores $b = 50$ experiences locally before sending them to replay buffer.

The numerical integration step in the training process is $\Delta T_{train} = 2s$. This is a trade-off between computational performance and integration accuracy of the mathematical model differential equations. However, the trained policies were evaluated in time-domain simulations using $\Delta T_{eval} = 0.1s$, as means of assuring that the coarseness of ΔT_{train} does not generate unsatisfactory policies when the fast-time simulations are executed with smaller integration step.

The distance between way-points $L_{segment}$ considers the trade-off mentioned in Section III-B, so that the desired path segment does not intersect margins considering the lowest radius of the bends r and the minimum channel width w compatible with real-world channels. The adopted values are compliant with the Harbour Approach Channels Design Guidelines [35]. For the navigation speed of $4m/s$, under moderate crosswind and cross current velocities, $w = 4Beam \approx 190m$. The minimum value for bend radius is $r = 5L_{oa} \approx 1400m$. Considering this scenario as the bottleneck, $L_{segment} = 650m$.

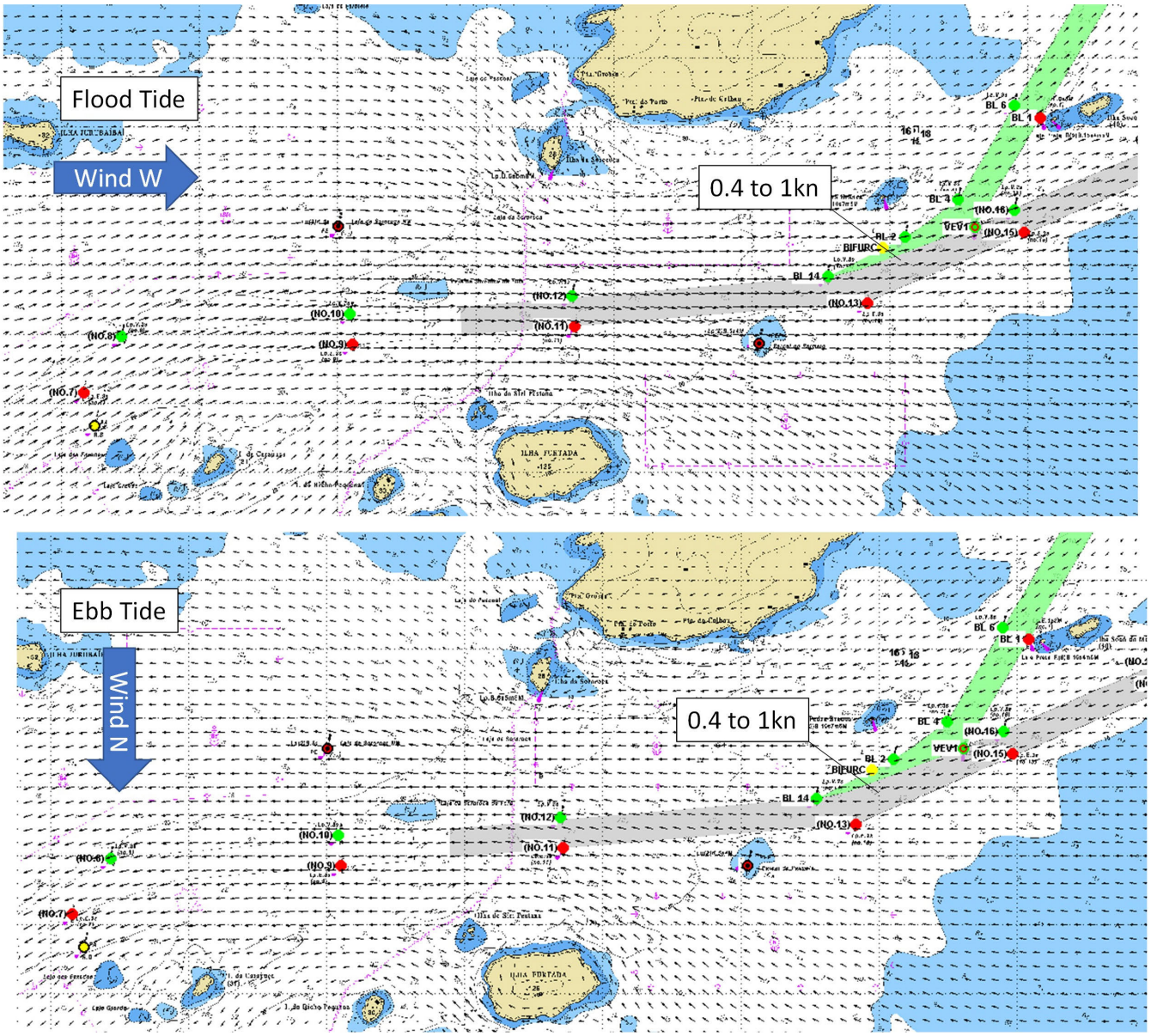

FIGURE 11. Environmental conditions - Current Vectors and Wind Direction: (up) Flood tide scenario; (down) Ebb tide scenario.

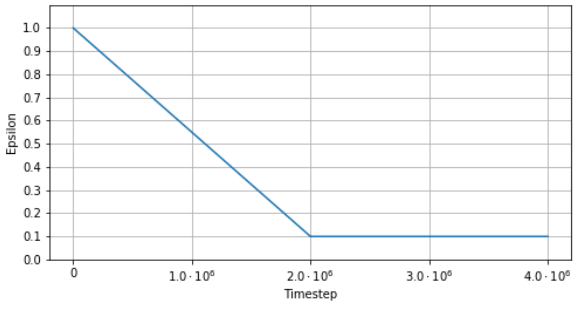

FIGURE 12. Exploration ϵ along training.

D. MULTIPLE SCENARIOS TRAINING

In this experiment, actors collected experiences in 4 different wind and current scenarios for Porto Sudeste simultaneously (E04, E08, F04 and F08, as shown in IV-B). These four scenarios were deemed representative of the phenomena we wish to study. This training was extended until the collection of 6.10^6 experience sets, and required less than 1h30 with the available computational resources. Three different trials were run both with APE-X-DQN and also with the PPO algorithm as a matter of comparison. The evolution of training is measured in terms of steps collected by the actors. In order to obtain a fair comparison, PPO trials were set with the same hyperparameters that are common to both algorithms, such as number of actors, batch size, steps collected per training iterations and learning rate. Other PPO specific parameters were maintained according to its original paper [10].

Figure 13 displays the average accumulated reward as training progresses. The more steps are collected and training iterations occur, the higher the average accumulated reward per episode grows until it stabilizes in a near-optimal

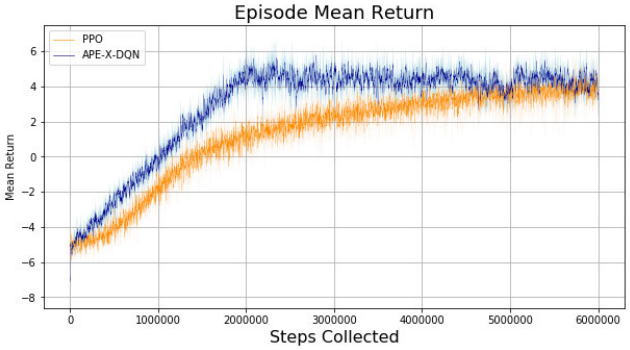

FIGURE 13. Average accumulated reward per episode (moving average from 100 episodes). Mean from 3 learning trials in darker orange and darker blue; standard deviation in shaded lighter orange and lighter blue.

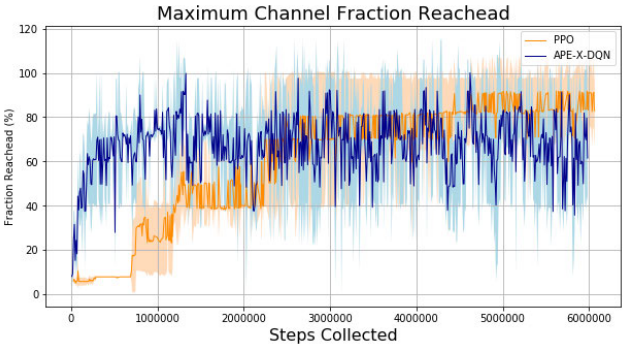

FIGURE 15. Fraction of channel reached by periodic evaluation of policy. Mean from 3 learning trials in darker orange and darker blue; Standard deviation in shaded lighter orange and lighter blue.

situation. Marginal oscillations are common as the episodes run over different portions of the channel. Although such oscillations occur for both algorithms, PPO displayed a slower convergence.

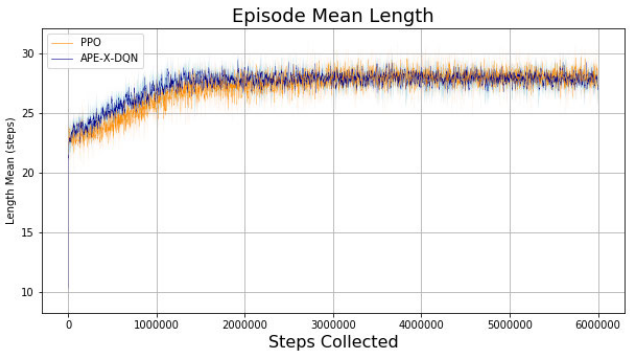

FIGURE 14. Average episode length (moving average from 100 episodes). Mean from 3 learning trials in darker orange and darker blue; Standard deviation in shaded lighter orange and lighter blue.

Figure 14 also conveys the training improvement as the average episode length grows, meaning that the vessels are navigating longer distances without collision.

As the desired trajectory is obtained by testing the policy in the full channel and the training episode is limited to the extent of two path segments, an evaluation is performed at every 10 training iterations. Figure 15 shows the maximum fraction of the channel obtained by the policy with exploration suspended in scenario E10. This scenario was not explored during training and is considered one of the most critical scenarios that occur in the channel area. Therefore, it was chosen to evaluate the generalization ability of the policy. The values were also averaged for 3 trials of each algorithm. The value shows high variance among trials as small differences in policy can make the vessel collide in early curves.

For practical purposes, the policy can be adopted as soon as the evaluation satisfies the established condition, which is 95% of the channel portion reached. The behavior can be extended to any other scenarios as well. Even with a somewhat oscillatory convergence, APE-X-DQN was able to produce a policy that generalized navigation to E10 scenario first.

The choice of APE-X-DQN is still more attractive in this domain not only for obtaining the desired policy faster, but also for estimating Q-values, as previously mentioned in Section II-A.

As an example of the policy performance, the trajectory obtained for training under condition E08 is depicted in Figure 16. The agent is able to maneuver in both curves, even under the effect of environmental forces. Note that the agent can anticipate the change in the orientation of the desired path segments to execute the curve with minimum overshoot.

Figure 17 shows the minimum distance between the vessel and the margins during navigation under Scenario E08. The agent could keep a safe distance from the margins during the complete maneuver. The minimum distance is 64m in the first curve at $t = 1200s$, which is acceptable (usual acceptance criteria adopted by pilots is distance larger than $0.5 - 1 \times Beam$).

The vessel's heading, course over ground (*cog*), and the channel centerline orientation are shown in Figure 18. The heading and *cog* are ahead of the channel orientation angle, showing the ability to anticipate the curve, as a result of the state-space model that contains information of the curvature ahead and distance to the orientation change. Consequently, the maneuver could be performed with almost no overshoot in the bends. The difference between *cog* and heading is the drift angle, required for compensating the perpendicular environmental forces. The value is less than 5° in the present case and is also employed by human pilots as a measurement of how difficult the course is.

The agent actions (rudder angle) are shown in Figure 19, for a 1000s time-window. The agent applies minimum positive rudder angle to keep the required drift angle for counteracting the environmental forces in straight segments ($t < 700s$ and $t > 1500s$). During the curve, there is a higher rudder action necessary to control the rate-of-turn.

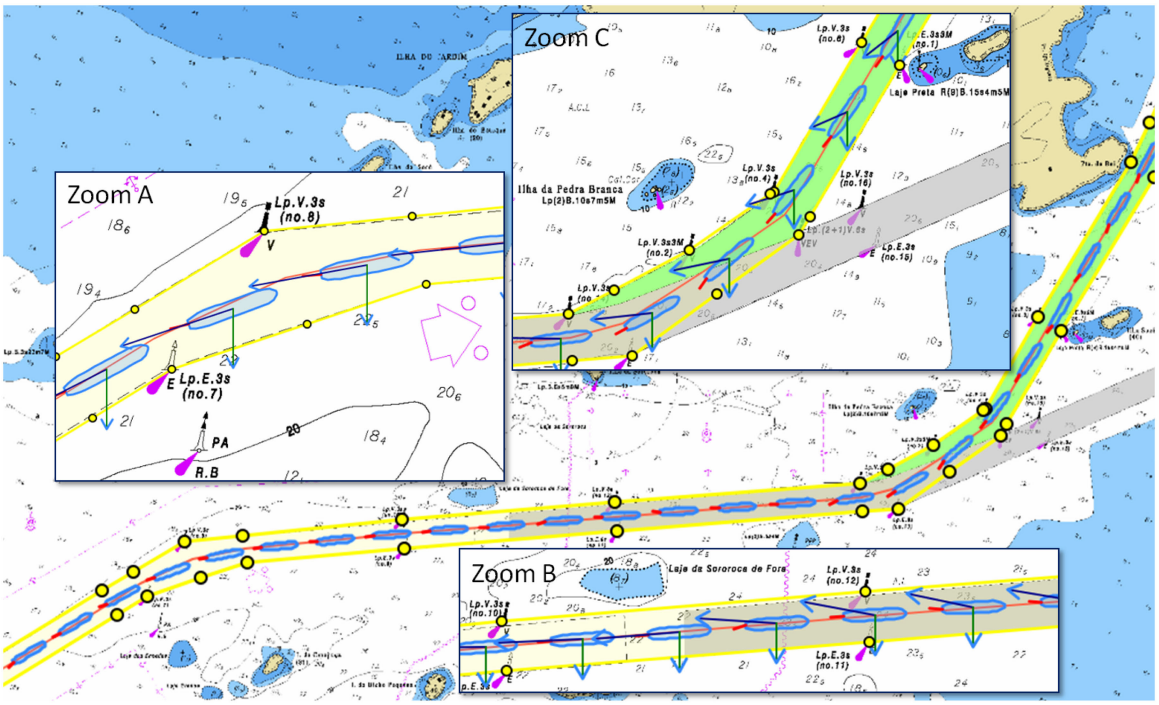

FIGURE 16. Trajectory obtained for condition E08 (green vectors=wind, blue vectors=current).

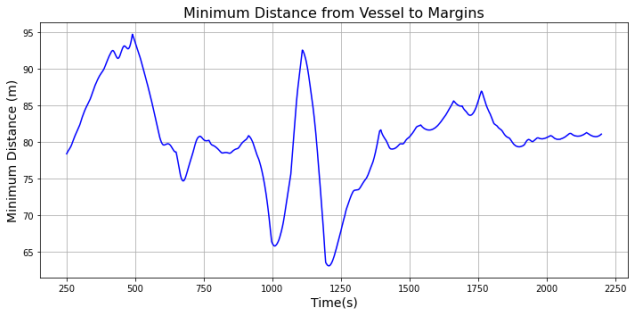

FIGURE 17. Minimum distance between vessel and margins for condition E08.

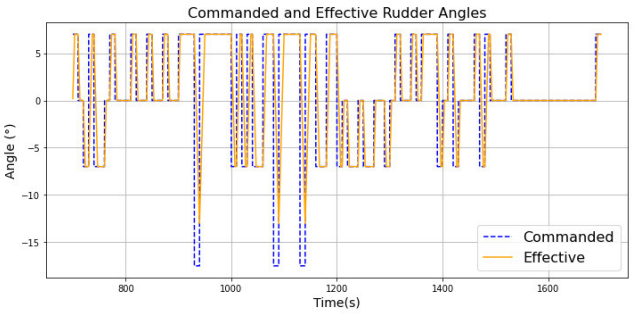

FIGURE 19. Rudder angles commanded by the agent and its effective value in Scenario E08.

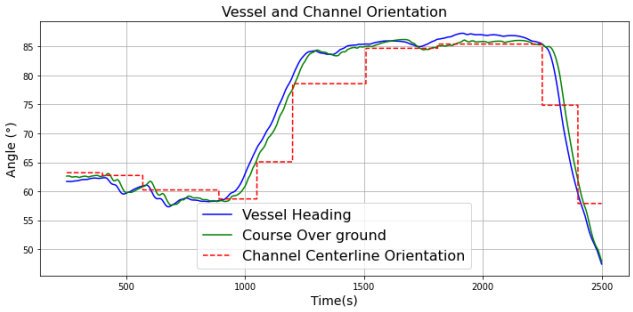

FIGURE 18. Vessel's heading and course over ground in Scenario E08.

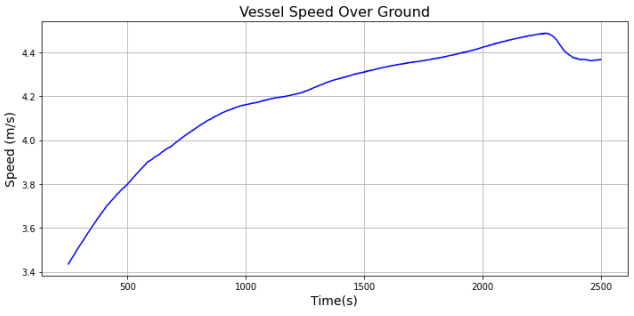

FIGURE 20. Speed over ground along the navigation in Scenario E08.

The frequency of rudder commands is similar to a human-controlled maneuver.

Although the maneuver is executed with a fixed propeller command, the speed over ground varies from $3.4m/s$ to $4.5m/s$, due to the variation of the longitudinal environmental forces and the speed loss after curves (Figure 20). This variation is sufficiently small such that the RL agent (that does not

take the speed as a state variable) could accurately control the vessel's maneuver. The experiences acquired during training enabled the agent to learn how to generalize the vessel's behavior in this speed interval.

The policy obtained was able to keep track of the desired path segments at the same time that environmental forces were compensated; that was achieved with rudder commands that closely resemble human piloting. As expected, the presence of a context variable and the distance to way-point provided to the policy improved the ability to anticipate curves without getting too close to margins.

E. ROBUSTNESS ANALYSIS

The policy trained in the four described scenarios simultaneously had improved extrapolation ability; the agent was able to successfully navigate in different scenarios from the ones used in training. Table 4 shows the new scenarios, with changes in wind conditions from E08. The agent was presented to those conditions and still could successfully navigate the complete channel (Figure 21).

TABLE 4. Wind changes from Scenario E08.

New Scenario	Changes from E08
E08WM20-	Wind velocity magnitude reduced in 20%
E08WM20+	Wind velocity magnitude increased in 20%
E08WD250	Wind velocity direction changed to 250°
E08WD290	Wind velocity direction changed to 290°

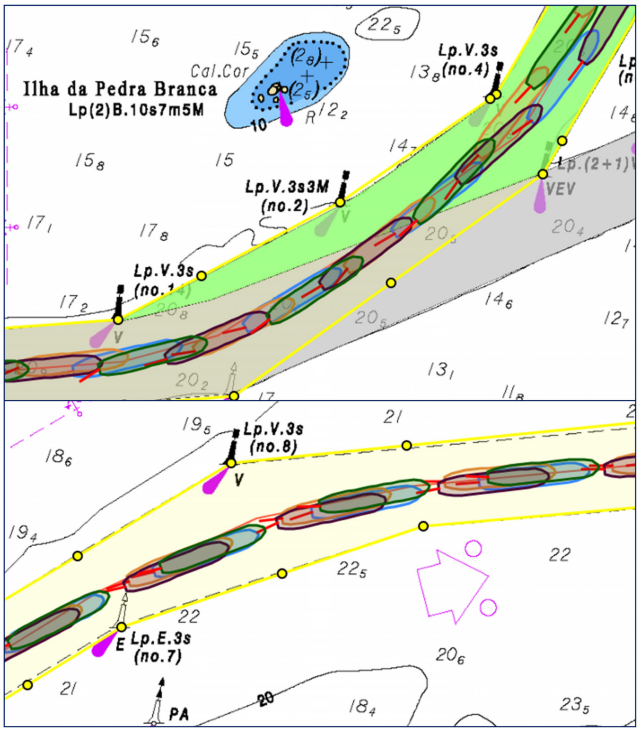

FIGURE 21. Policy trained in Scenarios E04,E08,F04,F08 and simultaneously tested in E08 with wind changes. Blue ship: Scenario E08WM20-; Orange ship: Scenario E08WM20+; Purple ship: E08WD250, Green ship: E08WD290.

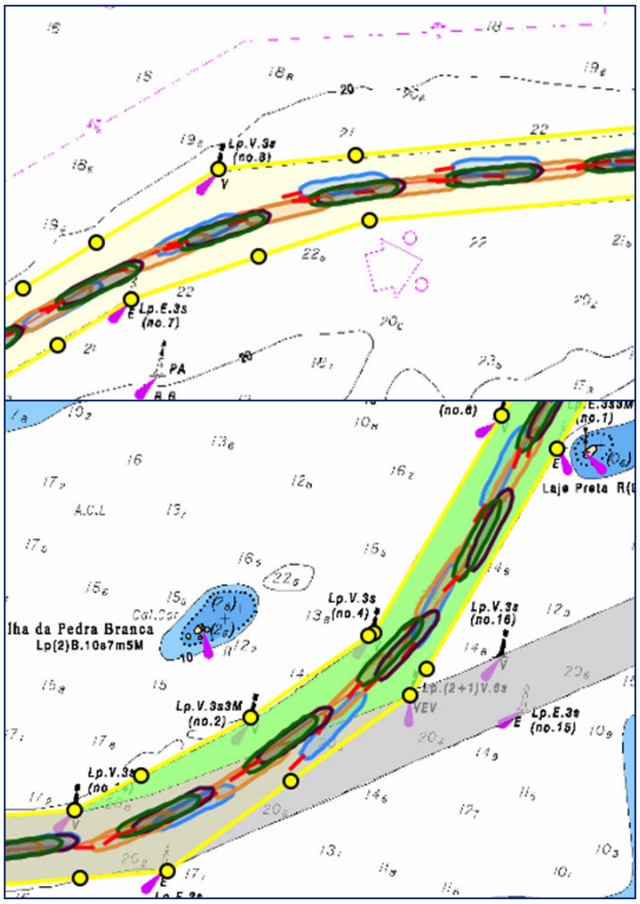

FIGURE 22. Policy trained in Scenarios E04,E08,F04,F08 and simultaneously tested in other scenarios. Blue ship: Scenario F05; Orange ship: Scenario E10; Purple ship: E05, Green ship: E06.

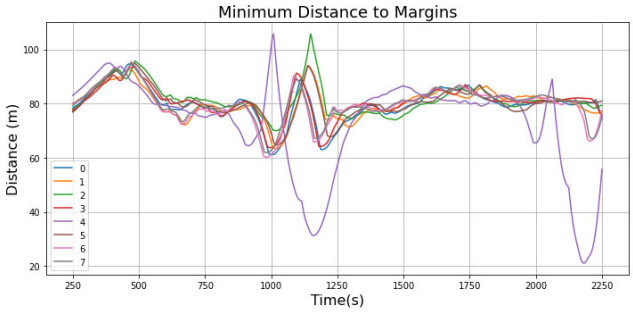

FIGURE 23. Distance to margins for policy trained with Scenarios E04,E08,F04,F08, and tested in other scenarios. (0-F05; 1-E10; 2-E05; 3-E06; 3-E08WM20-; 4-E08WM20+; 5-E08WD250; 6-E08WD290).

The robustness of the agent can also be accessed in scenarios with different current conditions from those used during training. Figure 22 demonstrates the behavior in Scenarios E05, E06, E10 and F05. For both ebb and flood currents, the agent executed the curves much better than in single scenario training and no collisions occurred.

Figure 23 depicts the distance from vessel to margins for all navigations in scenarios not included in training. All

trajectories succeed in reaching the end of the channel; note that the agent performed curves with safe distance to margins and small discrepancies in behavior amongst most scenarios.

Extrapolation to new conditions (robustness) was achieved because experiences under different scenarios were processed during training. That strategy prevented the neural network from overfitting to one environmental scenario.

V. CONCLUSION

We presented a solution for trajectory generation in fast-time simulations of navigation in restricted port channels. Our solution can handle narrow curves and environmental forces that might affect the vessel differently as channel orientation changes. We also employed human-like commands with discrete levels spaced by time intervals. Our solution is based on reinforcement learning and requires substantial parallel computing resources; we demonstrated its robustness by running through scenarios in which the controller had not been trained for.

Future work should include the final phase of ship berthing, which depends on commands to tugboats and propeller commands for velocity reduction. Although the present work focuses on fast-time simulations for project analysis, where obstacles are static, future work must also consider two-way channels with passing vessels, with further analysis on safety so that the model can be transferred to autonomous marine vehicles.

ACKNOWLEDGMENT

The authors would like to thank to Petrobras for supporting the development of the maneuvering simulator used in this work.

REFERENCES

[1] C. Chen, M. Vantorre, E. Lataire, M. Candries, K. Eloot, and A. J. Verwilligen, "Intelligent control strategies used in fast-time ship manoeuvring simulations," in *Proc. Marsim*, 2018, pp. 1–17.

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

[3] J. Amendola, E. A. Tannuri, F. G. Cozman, and A. H. R. Costa, "Port channel navigation subjected to environmental conditions using reinforcement learning," in *Proc. Int. Conf. Offshore Mech. Arctic Eng.*, vol. 58844. New York, NY, USA: American Society of Mechanical Engineers, 2019, pp. 1–10.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (Adaptive Computation and Machine Learning). Cambridge, MA, USA: MIT Press, 2018.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[6] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*. [Online]. Available: http://arxiv.org/abs/1511.05952

[7] F. Cruz, S. Magg, Y. Nagai, and S. Wermter, "Improving interactive reinforcement learning: What makes a good teacher?" *Connection Sci.*, vol. 30, no. 3, pp. 306–325, 2018.

[8] T. M. Moerland, J. Broekens, and C. M. Jonker, "Emotion in reinforcement learning agents and robots: A survey," *Mach. Learn.*, vol. 107, no. 2, pp. 443–480, Feb. 2018.

[9] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," 2018, *arXiv:1803.00933*. [Online]. Available: http://arxiv.org/abs/1803.00933

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347

[11] M. Stamenkovich, "An application of artificial neural networks for autonomous ship navigation through a channel," in *Proc. IEEE PLANS Position Location Navigat. Symp. Rec.*, Monterey, CA, USA, 1992, pp. 346–352.

[12] M. Lacki, "Reinforcement learning in ship handling," *TransNav, Int. J. Mar. Navigat. Saf. Sea Transp.*, vol. 2, no. 2, pp. 157–160, 2008.

[13] L. Zhang, L. Qiao, J. Chen, and W. Zhang, "Neural-network-based reinforcement learning control for path following of underactuated ships," in *Proc. 35th Chin. Control Conf. (CCC)*, Jul. 2016, pp. 5786–5791.

[14] A. B. Martinsen and A. M. Lekkas, "Straight-path following for underactuated marine vessels using deep reinforcement learning," *IFAC-PapersOnLine*, vol. 51, no. 29, pp. 329–334, 2018.

[15] A. B. Martinsen and A. M. Lekkas, "Curved path following with deep reinforcement learning: Results from three vessel models," in *Proc. OCEANS MTS/IEEE Charleston*, Oct. 2018, pp. 1–8.

[16] R. P. A. Rejaili and J. M. P. Figueiredo, "Deep reinforcement learning algorithms for ship navigation in restricted waters," *Mecatrone*, vol. 3, no. 1, pp. 1–10, 2018.

[17] C. Zhou, S. Gu, Y. Wen, Z. Du, C. Xiao, L. Huang, and M. Zhu, "The review unmanned surface vehicle path planning: Based on multi-modality constraint," *Ocean Eng.*, vol. 200, Mar. 2020, Art. no. 107043.

[18] K. Mitsubori, T. Kamio, and T. Tanaka, "Finding the shortest course of a ship based on reinforcement learning algorithm," *J. Jpn. Inst. Navigat.*, vol. 110, pp. 9–18, Mar. 2004.

[19] T. Kamio, K. Mitsubori, T. Tanaka, H. Fujisaka, and A. Haeiwa, "Effects of prior knowledge on multi-agent reinforcement leaning system to find courses of ships," *Austral. J. Intell. Inf. Process. Syst.*, vol. 12, no. 2, pp. 18–23, 2010.

[20] C. Wang, X. Zhang, R. Li, and P. Dong, "Path planning of maritime autonomous surface ships in unknown environment with reinforcement learning," in *Proc. Int. Conf. Cognit. Syst. Signal Process.* Singapore: Springer, 2018, pp. 127–137.

[21] C. Chen, X.-Q. Chen, F. Ma, X.-J. Zeng, and J. Wang, "A knowledge-free path planning approach for smart ships based on reinforcement learning," *Ocean Eng.*, vol. 189, Oct. 2019, Art. no. 106299.

[22] Y. Cheng and W. Zhang, "Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels," *Neurocomputing*, vol. 272, pp. 63–73, Jan. 2018.

[23] L. Zhao and M.-I. Roh, "COLREGs-compliant multiship collision avoidance based on deep reinforcement learning," *Ocean Eng.*, vol. 191, Nov. 2019, Art. no. 106436.

[24] E. Meyer, H. Robinson, A. Rasheed, and O. San, "Taming an autonomous surface vehicle for path following and collision avoidance using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 41466–41481, 2020.

[25] N. A. Q. Filho, M. Zimbres, and A. E. Tannuri, "Development and validation of a customizable DP system for a full bridge real time simulator," in *Proc. Int. Conf. Ocean, Offshore Arctic Eng. (OMAE)*, vol. 1A, 2014, pp. 1–11.

[26] E. A. Tannuri, F. Rateiro, C. H. Fucatu, M. D. Ferreira, Q. I. Masetti, and K. Nishimoto, "Modular mathematical model for a low-speed maneuvering simulator," in *Proc. 33rd Int. Conf. Ocean, Offshore Arctic Eng. (OMAE)*, San Franscisco, CA, USA, 2014, pp. 1–10.

[27] A. Bhatt, M. Argus, A. Amiranashvili, and T. Brox, "CrossNorm: Normalization for off-policy TD reinforcement learning," 2019, *arXiv:1902.05605*. [Online]. Available: https://arxiv.org/abs/1902.05605

[28] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1312–1320.

[29] J. Randløv and P. Alstrøm, "Learning to drive a bicycle using reinforcement learning and shaping," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 1998, pp. 463–471.

[30] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," 2017, *arXiv:1712.09381*. [Online]. Available: http://arxiv.org/abs/1712.09381

[31] S. S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.
[32] R. Hafner and M. Riedmiller, "Reinforcement learning in feedback control," *Mach. Learn.*, vol. 84, no. 1, pp. 137–169, Jul. 2011.
[33] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1–6.
[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations (ICLR)*, San Diego, CA, USA, Y. Bengio and Y. LeCun, Eds., May 2015. [Online]. Available: http://arxiv.org/abs/1412.6980
[35] M. McBride *et al.*, "Harbour approach channels—Design guidelines," PIANC, Gen. Secretariat Boulevard Roi Albert II, Brussels, Belgium, PIANC Rep. 121, Jan. 2014.

JOSÉ AMENDOLA received the degree in electrical engineering from the University of São Paulo, in 2013, where he is currently pursuing the master's degree with the TPN Laboratory. He is also a Software Developer with the TPN Laboratory, University of São Paulo. His current research interests include artificial intelligence, autonomous navigation, and intelligent control systems.

LUCAS S. MIURA is currently pursuing the bachelor's degree in automation and control engineering with the University of São Paulo. He is also a member of the University AI Turing Group. His current research interests include machine learning, data science, and software engineering.

ANNA H. REALI COSTA (Member, IEEE) received the Ph.D. degree in electrical engineering from the University of São Paulo (USP), Brazil. She has been a Guest Research Scientist with the Karlsruhe Institute of Technology (KIT), Germany, working on computer vision and intelligent mobile robots, with the Institute for Process Computing, Automation and Robotics (IPR), and with the FZI Research Center for Computer Science in Karlsruhe working on machine learning and computer vision. She has also been a Guest Researcher with Carnegie Mellon University, where she worked on planning, execution, and learning for multi-robot applications. She is currently a Full Professor with USP. She is also the Head of the Intelligent Techniques Research Laboratory, USP. Her research interests include machine learning, more specifically in reinforcement learning, autonomous agents, and transfer learning.

FÁBIO G. COZMAN received the engineering degree from the University of São Paulo (USP), Brazil, and the Ph.D. degree in robotics from Carnegie Mellon University, USA. He is currently a Full Professor with USP, where he works with probabilistic reasoning and machine learning. He has served, among other activities, as the Program Chair and a General Chair of the Conference on Uncertainty in Artificial Intelligence, an Area Chair of the International Joint Conference on Artificial Intelligence, an Associate Editor of the *Journal of Artificial Intelligence*, the *Journal of Artificial Intelligence Research*, and the *International Journal of Approximate Reasoning*, and the Chair of the Special Committee on Artificial Intelligence of the Brazilian Computer Society.

EDUARDO AOUN TANNURI (Member, IEEE) was born in São Paulo, Brazil, in 1976. He received the degree in mechatronics engineering and the Ph.D. degree in control from the University of São Paulo, Brazil, in 1998 and 2003, respectively.

He is currently a Full Professor with the Department of Mechatronics Engineering, University of São Paulo. He has published 188 articles in conferences and 38 in journals. His research interest includes non-linear dynamics and control, with applications in autonomous maritime vehicles and ship maneuverability. He has been a member of the International Towing Tank Conference (ITTC) Maneuvering Committee, since 2011. He received award of the Best Student Paper in the IFAC CAMS Conference, in 2001, and the National Prizes (Brazilian Navy Engineering Merit Award, in 2017, and the National Agency for Petroleum, Natural Gas and Biofuels Innovation Prize, in 2019).