Path planning for a maritime suface ship based on Deep Reinforcement Learning and weather data

Eva ARTUSI
MINES ParisTech - PSL Research University - Centre for research on Risks and Crises
NAVAL GROUP, Naval Research,
199 Avenue Pierre-Gilles de Gennes,
Ollioules, France
eva.artusi@mines-paristech.fr

Fabien CHAILLAN
NAVAL GROUP - Naval Research,
199 Avenue Pierre-Gilles de Gennes,
Ollioules, France
fabien.chaillan@naval-group.com

Aldo NAPOLI
MINES ParisTech - PSL Research University - Centre for research on Risks and Crises
Sophia-Antipolis, France
aldo.napoli@mines-paristech.fr

***Abstract*— Information analysis related to a ship and its environment is required in order to make the appropriate decisions during naval missions. However, human capabilities are no longer sufficient to reliably and rapidly process the massive amount of heterogeneous data collected by a huge lot of different sensors. That is the reason why Artificial Intelligence (AI) algorithms as decision support could help operators to choose the appropriate decisions during naval missions.**
This article offers a decision support model able to assist operators in predicting the path of a Maritime Surface Ship (MSS) in a dynamic environment by Deep Reinforcement Learning (DRL). Path planning of MSS in a dynamic environment is still challenging. Ocean disturbances are difficult to model thinly but collisions must be avoided. Thus, we suggest considering weather data and simplified mobile and static obstacles.

***Keywords*—Path planning, Ship, Deep Reinforcement Learning, Weather Data**

I. INTRODUCTION

A ship operator faces many challenges, including those associated with the dynamic environment and uncertainties in perception. To be able to make the right decision [1] he relies on information received from radar, AIS and other electronic equipment. Navigators decisions are based on electronic sources, but these decisions can be made by algorithms, which can help them optimize their mission's route [2].
The aim of this article is to offer a decision support able to assist operators in choosing the optimal situation-appropriate path during their mission in a dynamic environment.
Let us consider the scenario where the decision support model will help operators to navigate across the sea into a closed domain in order to reach as soon as possible a known goal location. In this case, the travel time should be minimized and will depend on changing environmental conditions, essentially wind and waves and on the maritime vehicle's ability to take these evolutions into account [3]. However, the disturbances caused by ocean are complex to model. To overcome this difficulty, weather data including ocean and wind information are used to train an intelligent agent to learn how to evolve within the environment.
This article is organized as follows: Section II reviews path planning methodology and DRL notions. Section III explains the suggested Maritime Surface Ship decision support based on DRL. Section IV shows simulation results and a comparison between the DRL and A*. Finally, the conclusion and future work are given in section V.

II. STATE OF THE ART

A. Path planning for MSS

For many years, the A* algorithm has been the dominant approach for the path planning problem. In [4], H. Erckens creates a boat which is capable of generating a persuasive path to a given destination and avoiding both static and dynamic obstacles based on the A* algorithm. Campbell, S [5] showed an unmanned surface vehicle (USV) real-time path planning method that was based on improved A* algorithm, which is integrated within a decision-making framework and combined with the Convention of the International Regulations for Preventing Collisions at Sea (COLREGs) [6]. The A* algorithm is a popular choice for path finding, however it relies on the design of the grid map, while the size and number of grids will directly affect the calculation speed and the accuracy of the algorithm. Thus, many intelligent algorithms, such as genetic algorithms, ant colony optimization algorithms or neural network algorithms [7] have been used in the autonomous path planning problem of unmanned ships. However, such intelligent algorithms are usually computationally intensive, and they are mainly used in offline global path planning or auxiliary decision making because they are difficult to use in the context of real-time ship action decision problems.
That is the reason why DRL has opened up new possibilities to solving path planning problems in a dynamic environment [8]. The essence of reinforcement learning (RL) is to train an agent to take an optimized action within its environment by maximizing some notion of cumulative reward. The learner is not told which actions to take, but instead must discover which actions yield the highest reward by trying them [9]. DRL is obtained by combining the advantages of Deep Learning (DL) and RL. For example, Woo [10] adopted a DRL technique for obstacle avoidance of an unmanned marine vehicle while Cheng [11] employs DQN [12] strategy for USV path planning and obstacle avoidance.

B. Deep Reinforcement Learning

RL [9] is modeled as a Markov Decision Process (MDP), where an agent during the training has its state s_t updated at every time, taking an action a_t based on a policy π and then receiving a reward $r(s_t, a_t)$ provided by the environment. As

978-0-692-93559-0 ©2021 MTS

a result of the action a_t, the current state s_t may change according to the transition probability model $p(s_{t+1}|s_t, a_t)$. **Fig 1** shows the general reinforcement learning schematic. The agent's goal is to learn an optimal policy π^* to maximize its expected sum of rewards R_t :

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (1)$$

$0 \leq \gamma \leq 1$ is the discount factor, which weights the potential fture errors.

A policy can be evaluated by using either a state value function $V_\pi(s_t)$ defined as the accumulated discounted reward expectation; or an action value function $Q_\pi(s_t, a_t)$ defined as a value function for the specific state and action pair.

$$V_\pi(s_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | \, s_t = s \right] \quad (2)$$

$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | \, s_t = s \, , a_t = a \right] \quad (3)$$

Therefore, the optimal policy π^*satisfies the following conditions:

$$\pi^* = \underset{\pi \in [0,1]}{\operatorname{argmax}} [V_\pi(s_t)] = \underset{\pi \in [0,1]}{\operatorname{argmax}} [Q_\pi(s_t, a_t)] \quad (4)$$

To resolve RL problem, a table is used for storing the values of $V_\pi(s_t)$ and $Q_\pi(s_t, a_t)$. However, a simple table is not sufficient to handle the combinatorial of real-world problems. Instead of using a table as an approximator of the value functions, neural networks are used to tackle this issue. This method called DRL is illustrated on the **Error! Reference source not found.**, where the agent's policy will be approximated by a neural network.

III. MARITIME SURFACE SHIP MODEL FOR DRL TRAINING

In this section, we first introduce the MSS kinematic motion that predominately regulates the state transition within the MDP. Then in order to address the path planning issue, we suggest formulating the DRL framework, where the state, actions and reward function must be pre-designed so that the agent can learn the policy. The Proximal Policy Gradient algorithm [13] is used to learn the policy. Once the training process will be completed, the MSS could automatically navigate in new situations and arrive at the fixed goal area while avoiding any collision.

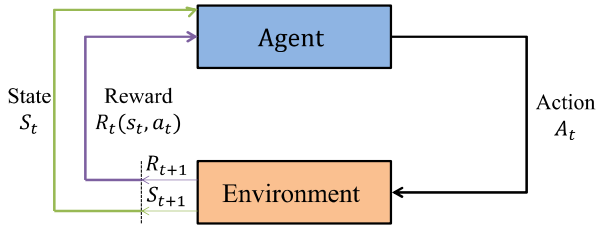

Fig 1 : The reinforcement learning schematic

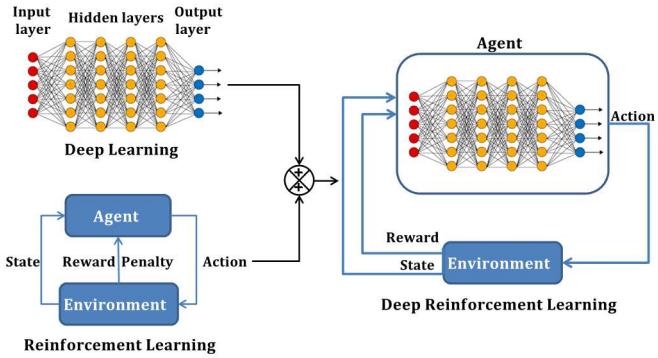

Fig 2 : DRL: the combination between RL and neural networks

A. The kinematic motion of MSS and environmental forces

In this study, a simplified three-degree-of-freedom (3-DOF) vessel kinematic model is used to describe the MSS motions in the horizontal plane. The rigid body kinematic equation is

$$\dot{\eta} = R(\psi) v \quad (5)$$

Where $\eta = [x, y, \psi]^T$ represents the earth-fixed position and heading angle, $v = [u, v, r]^T$ represents the vessel-fixed velocities with r constant in our case. $R(\psi)$ is the rotation matrix from the earth-fixed frame to the vessel-fixed frame.

$$R(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6)$$

And the ship speed is $V = \sqrt{u^2 + v^2}$

The kinematic motion model of MSS will be incorporated into the designed MDP. **Fig 3** illustrates the environmental components interacting with the ship and represented by two parameters: wind and wave forces, where β_{wind}, β_{wave} are wind and wave direction in an earth-fixed reference, and $\alpha_{r_{wind}}$, $\alpha_{r_{wave}}$ are the relative wind and wave direction in vessel-fixed frame and calculated as:

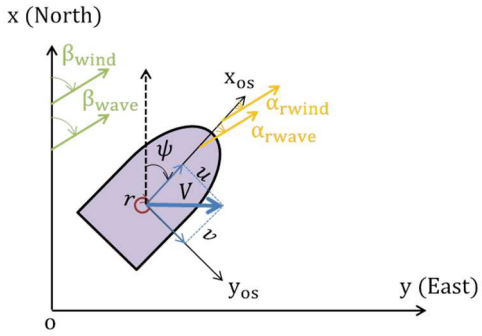

Fig 3 : The ship kinematic parameters

$$\alpha_{r_{wind}} = \psi - \beta_{wind}$$
$$\alpha_{r_{wave}} = \psi - \beta_{wave} \quad (7)$$

Wind forces can be described by the longitudinal wind force, $F_{x_{wind}}$, lateral wind force, F_{ywind}, and wind moment M_{zwind}. We assume that lateral wind force and wind moment are neglected and longitudinal wind force can be computed as [14]:

$$F_{x_{wind}} = \frac{1}{2}\rho_{air} A_T C_{wx}(\alpha_{r_{wind}}) V_{r_{wind}}^2 \quad (8)$$

With ρ_{air} , the air density, A_T, the transversal projected wind area,C_{wx}, wind load coefficient, $V_{r_{wind}}$, the relative wind speed.

We only consider the second-order wave drift force and neglect the wave force moment.

$$F_{x_{wave}} = \frac{1}{2}\rho_{ocean} h^2 g L \, C_{wavex}(\alpha_{rwave})$$
$$F_{y_{wave}} = \frac{1}{2}\rho_{ocean} h^2 g L \, C_{wavey}(\alpha_{rwave}) \quad (9)$$

With ρ_{ocean}, the ocean density, h, the total swell height, g, the gravitational force, L, the ship length, C_{wavex}, C_{wavey}, the coefficients of the second-order wave drift force.

B. Surge speed computing

It will be assumed that minimizing the travel time of the MSS is equivalent to maximizing its surge speed depending on a number of MSS parameters:

$$V = \frac{P(V)\eta_p}{\mathrm{RFM}(V)} \quad (10)$$

Where V is the surge speed in knots, $P(V)$ the required propelling power, η_p the propulsive efficiency and RFM the resistance to forward movement. For this study we consider a simplified RFM model, where only the four most influential environmental parameters and headwinds affects have been considered:

$$RFM = \underbrace{R_o(V) + \nu(V, \alpha_{r_{wave}}, h) R_w(V, h)}_{\text{Wave forces}} + F_{x_{wind}}(V_{r_{wind}}, \alpha_{r_{wind}}) \quad (11)$$

R_0 ,the resistance to forward movement without wind and swell (N).

R_w,the resistance to forward movement for a given swell in front of the ship (N).

v, wave angle correction factor (rad).

$F_{x_{wind}}$, wind force along the ship's longitudinal axis (N).

h, the height of total swell (m).

V the MSS speed assumed to be equal to the surge speed (m/s).

The numeric value of V in (10) is obtained by way of a fixed-point iteration.

C. Environment establishment

Environment will be modeled as a grid representing an ocean area. Map data are expressed in latitude and longitude according to a common standard such as WGS84. Ocean area is defined by $[\mathrm{lat}_{min}, \mathrm{lat}_{max}] \times [\mathrm{lon}_{min}, \mathrm{lon}_{max}]$ and illustrated on **Fig 4**.

The mesh is regularly meshed with Δlat and Δlon. The **Tab 1** reviews the fixed parameters values used in our study. As in this case weather data show a coarser discretized ($\Delta\mathrm{lat}_{\mathrm{meteo}}, \Delta\mathrm{lon}_{\mathrm{meteo}}$), we assume that a weather data value will be set on the grid by interpolation with the nearest neighbour.

Since the ocean area is supposed to be compact enough to be locally Euclidean, the MSS motion is supposed to be uniformly rectilinear and its coordinates first expressed in distance will be converted into latitude and longitude.

Mobile obstacles motions are described in exactly the same way as the MSS motion except that weather data do not affect their motions.

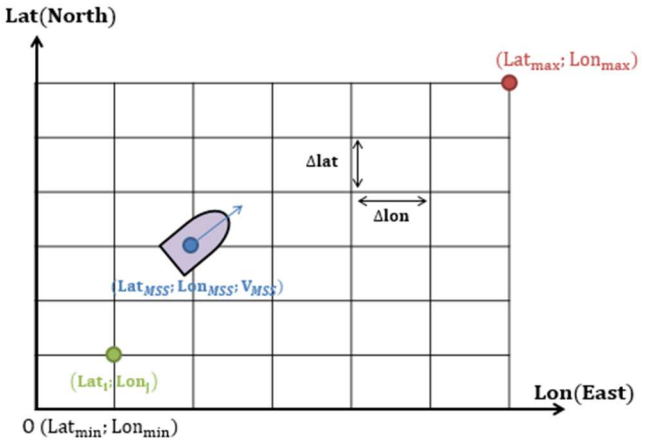

Fig 4 : Definition of the ocean area

Tab 1 : Fixed parameters values

Variables	Value
lat_{min}	User value
lat_{max}	$\text{lat}_{min} + 1.5°$
lon_{min}	User value
lon_{max}	$\text{lon}_{min} + 1.5°$
Δlat	0.001°
Δlon	0.001°
$\Delta\text{lat}_{\text{meteo}}$	0.5°
$\Delta\text{lon}_{\text{meteo}}$	0.5°

D. State space

For this study, the state space is defined as a vector including MSS and obstacles information. The first row of the vector will be composed of the MSS position coordinate (lat_{MSS}, lon_{MSS}), its heading angle, its surge speed and the Euclidean distance between its position and the goal point. The following rows will be composed of obstacles position coordinates, their heading angles, their speed and Euclidean distance between their position and the MSS position.
On this ocean area, let "S" be the countable finite set of possible states of the agent (ie: MSS):

$$S = \begin{bmatrix} lat_1 & lon_1 & \psi_1 & V_1 & d_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ lat_N & lon_N & \psi_N & V_N & d_N \end{bmatrix} \quad (12)$$

With N the number of mobile and static obstacles.

E. Action space

It is assumed MSS will maintain its surge speed during the navigation, so actions taken by the agent (ie: MSS) are the heading angles in degree. The action space is defined as a set of 5 discrete actions:

$$A = [-30°, -10°, 0°, 10°, 30°] \quad (13)$$

Here A is the finite set of possible actions and $a \in$ A is an allowable action.
According to the defined state and action space, the MSS state transition with a given action a by adhering to the kinematics of MSS can be calculated as, $\forall t \in [0; T]$:

$$\begin{cases} \psi' = \psi + a \\ x_{MSS}(t') = x_{Mss}(t) + V(t')\cos(\psi')t' \\ y_{MSS}(t') = y_{MSS}(t) + V(t')\sin(\psi')\,t' \end{cases} \quad (14)$$

$x_{MSS}(t')$ and $y_{MSS}(t')$ are converted in $lat_{MSS}(t')$ and $lon_{MSS}(t')$.
The obstacles state transitions will be described as well by the equation (14) except that heading angles and speeds will be at each time step pseudo-randomly generated.

F. Reward function

The reward function plays an essential role in the MSS DRL system [15]. It should be designed to make an agent learn a desired behavior and complete defined tasks as expected. In our case the MSS has to optimize its travel time by using the roads leading to the maximum surge speed, limiting the total number of displacements in space and avoiding all obstacles. The total reward will be decomposed into a set of sub-rewards functions in order to take objectives into account and avoid collisions [8].

Reaching the terminal state:

The goal reward is designed to encourage the MSS to reach the goal area. It will only be given when the MSS manages to reach this area. The distance between the MSS current position and the goal position is given by:

$$d(MSS, goal) = \sqrt{\left(lat_{MSS} - lat_{goal}\right)^2 + (lon_{MSS} - lon_{goal})^2} \quad (15)$$

So the goal reward is defined by:

$$R_{goal} = \begin{cases} \lambda_{goal}\ if\ d(MSS, goal) < r_{goal} \\ 0\ ohterwise \end{cases} \quad (16)$$

Since the goal achievement is one of the most critical criteria of completing a mission, with λ_{goal} a large positive value and r_{goal} the goal area radius centered in $(\text{lat}_{\text{goal}}, \text{lon}_{\text{goal}})$.

Minimizing the travel time:

The distance reward function will be defined in order to find the shortest path to destination as:

$$R_{distance} = -\lambda_{distance} d(MSS, goal) \quad (17)$$

With $\lambda_{distance} > 0$, it has been designed in a way that, the closer the MSS and the destination, the less punishment is imposed. This distance reward will encourage the MSS to choose areas where its surge speed will be maximum for reaching the goal area as soon as possible.

Near-collision and collision avoidance:

The collision avoidance reward function is designed to prevent near-collisions and collisions with both static and mobile obstacles. Static and mobile obstacles are pseudo-randomly generated at each episode. Note that the COLREGs are not taken into account.
The ship and mobile obstacles domains are simplified by two circles of radii $r_{obstacle}$ and r_{MSS} as illustrated on **Fig 5.**
The radius r_{MSS} can be considered as the minimum passing distance between the MSS and mobiles obstacles and is equal to $3.6L$, where L is the length ship. $3.6L$ is extracted from the Fuji model [16]. The radius $r_{obstacle}$ is supposed to be the same for each of all obstacles.

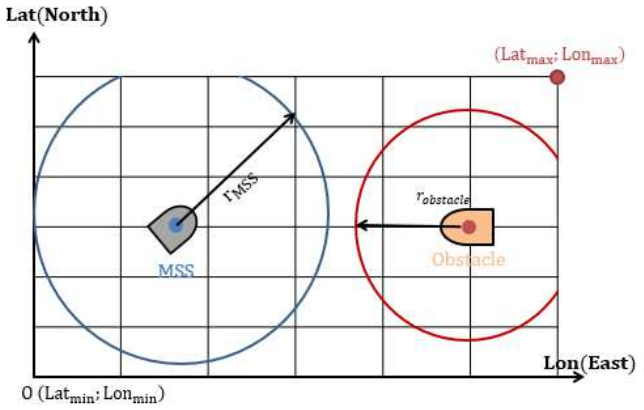

Fig 5 : Illustration of $\boldsymbol{r_{obstacle}}$ and $\boldsymbol{r_{MSS}}$

The distance between MSS current position and the obstacle position is given by:

$$d(MSS, obst) = \sqrt{(lat_{MSS} - lat_{obst})^2 + (lon_{MSS} - lon_{obst})^2} \quad (18)$$

So the collision avoidance reward is defined by:

$$R_{collision} = \begin{cases} -\lambda_{collision} \; if \; d(MSS, obst) < r \\ 0 \; otherwise \end{cases} \quad (19)$$

Where $\lambda_{\text{obstacle}}$ is a constant to heavily 'punish' the agent colliding or near-colliding with obstacles and 'suggest' to the vessel not to take this action if similar scenario would occur in future training stages and $r = \max_{r_{MSS}, r_{obstacle}} [r_{MSS}, r_{obstacle}]$.

During the simulation, the reward value is defined by the following rules:

(1) If $\text{Lat}_{\text{MSS}} < Lat_{inf}$ or $\text{Lon}_{\text{MSS}} < Lon_{inf}$, the ship is out of the grid and the agent will be penalized. The current episodes stop and a new one is launched.
(2) If $d(MSS, obst) < r$, the ship has near-collided or collided with an obstacle so $R_{total} = R_{collision}$. The current episodes stop and a new one is launched.
(3) If $d(MSS, goal) < r_{goal}$ the ship has reached the goal area,$\text{R}_{\text{total}} = \text{R}_{\text{goal}}$. The current episodes stop and a new one is launched.
(4) Otherwise, the ship is considered sailing properly in the simulation world, $R_{total} = R_{distance}$ and the learning process continues.

The **Fig 6** illustrates the whole DRL framework.

G. Proximal Policy Optimization algorithm (PPO)

For this study, PPO algorithm [13] was chosen to train our agent, which is one of the most popular DRL methods because of its using facility and good performance. PPO is an 'on-policy algorithm' where policy is updated explicitly: during the training, a small batch of experiences interacting with the environment is collected and used to update its policy.

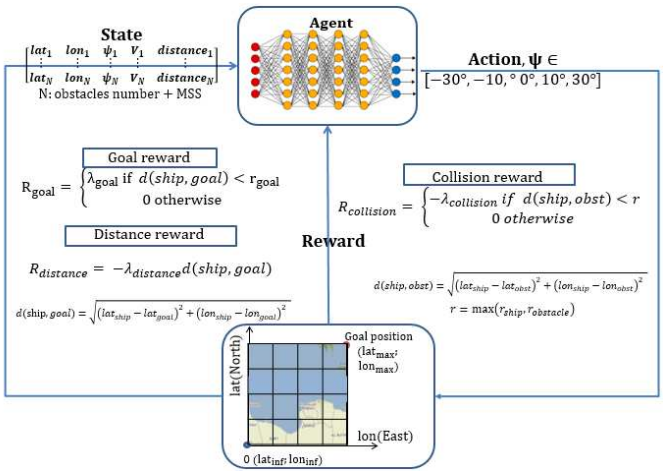

Fig 6 : The DRL framework

Once the policy is updated with this batch, the experiences are thrown away and a newer batch is collected with the newly updated policy. We chose PPO because it is well suited for discrete actions and multi-processing as it provides faster convergence and performance rate. As well it often has very poor sample efficiency taking millions or billions of time steps to learn simple tasks.

IV. EXPERIMENTS AND RESULTS

A. Weather data and scenario

The ERA5 reanalysis-dataset from ECMWF was used as weather data [16]. Reanalysis combines observations into globally complete fields using the laws of physics with the method of data assimilation. These data have then been spatially resampled in order to fit with a regular Lat-Lon grid with a space resolution of 0.5 degrees. 0.5 degrees is the correspondence to about 56 km. The data was therefore available on its three-hourly resolution. The latitude (**lats**), the longitude (**longs**), the 10m wind speed (**wind**), the date of the forecast (**time**), the air temperature (**t2m**), the wind direction (**mdww**), the swell direction (**mwd**), the swell's height (**shts**) and the sea state (**SS**) are presented in our use case. All these data are expressed in SI unit, except the sea state which stands without unit and is represented as a grid where the MSS will evolve. The NaN values represent lands (islands or coasts). Example of the swell's height repartition is given in **Fig 7**.

Our agent will be trained on different episodes. For each training episode, an area equal to $[lat_{min,}\, lat_{max}] \times [lon_{min}, lon_{max}]$ (please note the fixed parameters in **Tab 1**) will be defined, where lat_{min} and lon_{min} are pseudo randomly chosen and weather data corresponding to the area will be updated. In order to make the agent training easier, the area $[lat_{min,}\, lat_{max}] \times [lon_{min}, lon_{max}]$ will be transformed into relative coordinates and will always be defined by $[0°, 1.5°] \times [0°, 1.5°]$.

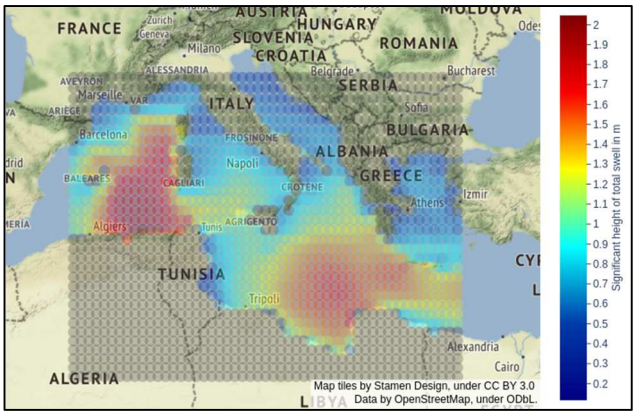

Fig 7 : Repartition of the swell's height (shts) in part of the Mediterranean Sea

The MSS begins from a starting position, which is pseudo randomly generated for each new episode and has a goal to reach. This goal is an area centered in(lat_{max}, lon_{max}), which is always the same during training episodes.

Besides, for each episode, new obstacles positions and their speed are pseudo randomly generated. The obstacle number on the grid is always 60. Each time step in the simulator is equivalent to 8 minutes because the length grid is $1.5° \times 1.5°$ equivalent to $168km \times 168km$. Maximum number of steps per episode is set to 200 steps equivalent to about 27 hours. The episode ends if the goal is reached, if an obstacle is near collided or collided, if the agent (MSS) gets out of the grid or if the maximum number of steps is reached. During the training, at each time step, mobile obstacles change their speeds and their heading angles pseudo-randomly. The MSS maximum surge speed is set to 26 knots.

B. Tools, libraries and training

The implementation uses the RL framework provided by the Python library **OpenAI Gym** [17] and PPO is extracted from **Stable Baselines** [18]**,** called PPO2. The PPO2 algorithm was trained with 16 CPU on 3,000,000 time steps, with each step generating an experience consisting of current observed state, the action selected, the corresponding reward received and the subsequent next state. The policy used is a multi layer perceptron composed of 2 layers with 64 neurons. The learning rate is 0.00025, the entropy coefficient 0.01 and the discount factor 0.99.

An ϵ greedy policy is followed during the training. With this strategy, an exploration rate ϵ is set initially to 1. This exploration rate is the probability that our agent will explore the environment rather than exploit it. With $\epsilon = 1$, it is 100% certain that the agent will start out by exploring the environment.

As the agent learns more about the environment, at the beginning of each new episode, ϵ will decay by some rate that is set so that the likelihood of exploration becomes less and less probable (as the agent learns more and more about the environment). The agent will become 'greedy' in terms of exploiting the environment once it has had the opportunity of exploring and learning more about it.

C. Simulation results

The specific training process was first analysed in order to ensuring that the agent learned well. **Fig 8** shows the episode reward as a function of time steps. The PPO agent initially learned a policy leading to an episode reward of −2, suggesting the agent has gained negative experiences to a maximum degree. From this point onward, the agent managed to improve the MSS control policy gradually, and the episode reward converged to approximately 5. The training was terminated at 3, 000,000-time steps. After the training, different simulations were realized on real ocean areas with weather data extracted from ECMWF and updated every three hours. Error! Reference source not found. illustrates four MSS predicted path controlled by the PPO policy. The colorbar represents surge speed values in knots. The operator initializes the starting point and the ocean area. The goal area is always the same. The four areas are different as well as are the obstacles coordinates. Since we cannot represent mobile obstacles on figures, only static obstacles are represented in white. Every three hours, the weather data are updated during the episode. Choosing heading angles and avoiding static obstacles, the MSS is able to reach the goal area while maximizing its surge speed all along its patt. The anti-collision performance was evaluated on 500 different episodes pseudo-randomly generated. The meant success obtained is 93%.

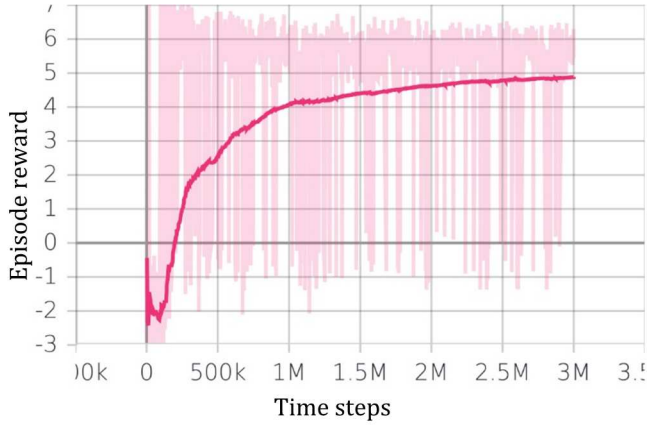

Fig 8 : MSS path planning training process

D. Path planning comparison between A and the decision support model*

In the same scenarii as previous, the decision support model (DSM) and A* algorithms will be compared according to the algorithm run time and the distance travelled. Many kinds of improved A* have been proposed in the applications of vessels under different scenarii but for this first comparison, only the original A* is used and the following assumptions were made:

- Only static obstacles have been taken into account.

- Weather data are constant on the maritime area in order to implement a simple A* cost function.
- The Euclidean distance called $d(a,b)$ is used as the function heuristic of A*. This distance allows eight disrecte actions: up, down, right, left and the diagonals.
- The previous DRL framework of the decision support model is modified. The agent action space will be defined as a set of 8 discrete actions: up, down, right, left and the diagonals.

A* and the decision support model were compared on two use cases. For each use case the maritime area was the same, defined as $[33°, 34.5°] \times [15.5°, 17°]$ with thirty static obstacles for the first use case and sixty for the second. Static obstacles and start point were pseudo-randomly generated and as weather data are constant all along the travel, the MSS surge speed was constant. Results are shown in **Tab 2**, where the average run time and the average distance travelled were computed for ten differents scenarios in each use case.

Tab 2: Comparison between A* and the DSM

Static obstacles number	Algorithm	Average run time (s)	Average distance travelled (km)
30	A*	3.80 ± 0.78	130 ± 29
	DSM (PPO2)	1.67 ± 0.19	126.5 ± 31
60	A*	4.25 ± 1.04	133.5 ± 32.8
	DSM (PPO2)	1.91 ± 0.10	130.5 ± 33.5

We notice that the decision support model with PPO2 agent (i.e: algorithm) is much faster than A* in the two use cases. The difference between the average run time increases when the static obstacles number rises, indeed the difference is 2.13 s for thirty static obstacles and 2.34 s for sixty static obstacles. Besides the standard deviation of run time increases for A* and decreases for the DSM. We can assume that the gap will be larger if the scenarii are more complex. This gap is not surprising because the computer resources to be mobilised are very different. The DRL agent can take a long time to train but once trained it is able to predict a path in a few seconds and without mobilising any IT resources. On the other hand A* will take several minutes and seconds to find a solution and requires a lot of IT resources each time. About the average distance travelled the gap is not significant and roughly constant, but it could be much larger with the introduction of mobile obstacles. Therefore, with the first comparison, our decision support model with PPO2 agent is able to assist operators to choose the optimal path situation-appropriate in a dynamic environment.

V. CONCLUSION

This research offers a decision support model aimed to assist operators in predicting the optimal path situation the most appropriate considering weather data. Based on the PPO2 algorithm, a MSS path prediction model was realized providing heading angles recommendation to operators. The agent was previously trained with different geographical areas weather data with pseudo-random starting points and pseudo-random mobile obstacles. The MSS behavior and the mobile obstacles were simplified in this simulation. The anti-collision performance was evaluated and the system was validated on a simple comparison with A* for this first study. The next step will be to improve the validation system to compare A* and the decision support model with mobile obstacle and enabling the simulation system to evolve towards a more realistic environment, where maritime traffic and COLREGs will also be considered.

REFERENCES

[1] Z. Pietrzykowski, P. Hatłas, A. Wójcik, et P. Wołejsza, « Subontology of communication in the automation of negotiating processes in maritime navigation », *Zesz. Nauk. Akad. Morskiej W Szczecinie*, vol. nr 46 (118).

[2] E. Kulbiej et P. Wołejsza, « Naval Artificial Intelligence », juin 2017, p. 291-297.

[3] C. Huang, K. Yin, et L. Liu, « Learning Partially Structured Environmental Dynamics for Marine Robotic Navigation », *ArXiv180304057 Cs*, mars 2018.

[4] H. Erckens, G.-A. Beusser, C. Pradalier, et R. Y. Siegwart, « Avalon », *IEEE Robot. Autom. Mag.*, vol. 17, nº 1, p. 45-54, mars 2010.

[5] S. Campbell, W. Naeem, et G. W. Irwin, « A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres », *Annu. Rev. Control*, vol. 36, nº 2, p. 267-283, déc. 2012.

[6] Z. Luman et M.-I. Roh, « COLREGs-compliant multiship collision avoidance based on deep reinforcement learning », *Ocean Eng.*, vol. 191, oct. 2019.

[7] L. Ran, Y. Zhang, Q. Zhang, et T. Yang, « Convolutional Neural Network-Based Robot Navigation Using Uncalibrated Spherical Images », *Sensors*, vol. 17, nº 6, juin 2017.

[8] X. Zhou, P. Wu, H. Zhang, W. Guo, et Y. Liu, « Learn to Navigate: Cooperative Path Planning for Unmanned Surface Vehicles Using Deep Reinforcement Learning », *IEEE Access*, vol. 7, p. 165262-165278, 2019.

[9] R. S. Sutton et A. G. Barto, *Reinforcement learning: an introduction*, Second edition. Cambridge, Massachusetts: The MIT Press, 2018.

[10] J. Woo, C. Yu, et N. Kim, « Deep reinforcement learning-based controller for path following of an unmanned surface vehicle », *Ocean Eng.*, vol. 183, p. 155-166, juill. 2019.

[11] Y. Cheng et W. Zhang, « Concise Deep Reinforcement Learning Obstacle Avoidance for Underactuated Unmanned Marine Vessels », *Neurocomput.*, vol. 272, nº C, p. 63-73, janv. 2018.

[12] V. Mnih *et al.*, « Playing Atari with Deep Reinforcement Learning », déc. 2013.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, et O. Klimov, « Proximal Policy Optimization Algorithms », *ArXiv170706347 Cs*, août 2017.

[14] « OffshoreHydromechanics_Journee_Massie.pdf ». https://ocw.tudelft.nl/wpcontent/uploads/OffshoreHydromechanics_Journee_Massie.pdf

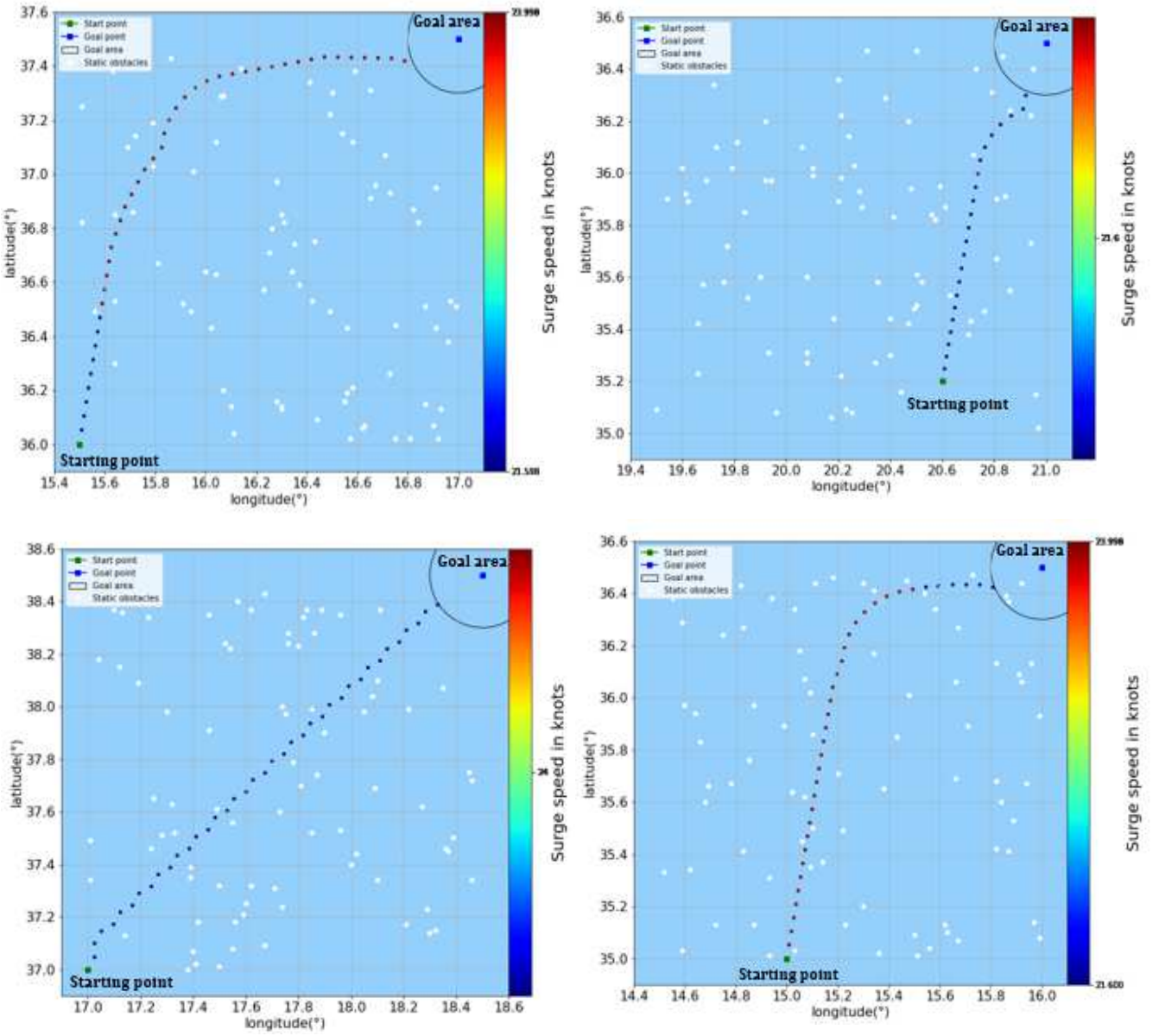

Fig 9: Four MSS predicted path controlled by the PPO policy.

[15] X. Geng, Y. Wang, P. Wang, et B. Zhang, « Motion Plan of Maritime Autonomous Surface Ships by Dynamic Programming for Collision Avoidance and Speed Optimization », *Sensors*, vol. 19, p. 434, janv. 2019.

[16] Meteorological dataset: https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-single-levels-monthly-means?tab=form

[17] Brockman, Cheung, Pettersson, Schneider, Schulman, Tang, and Zaremba, ‘‘OpenAI gym,’’ CoRR, vol. abs/1606.01540, 2016.

[18] Hill, Raffin, Ernestus, Gleave, Kanervisto, Traore, Dhariwal, Hesse, Klimov, Nichol, Plappert, Radford, Schulman, Sidor, and Wu. (2018). Stable baselines.