DRL-Based Thruster Fault Recovery for Unmanned Underwater Vehicles

Katell Lagattu, Gilles Le Chenadec, Eva Artusi, Paulo Santos, Karl Sammut, Benoit Clement

DRL-Based Thruster Fault Recovery for Unmanned Underwater Vehicles

Katell Lagattu[1,2,3,4], Gilles Le Chenadec[1], Eva Artusi[2], Paulo E. Santos[3,4], Karl Sammut[3,4], Benoit Clement[3,4]

***Abstract*— Thruster faults are one of the most common malfunctions encountered during Unmanned Underwater Vehicle (UUV) missions. This type of fault can lead to unwanted behaviour and jeopardise the UUV mission. Successful thruster fault management depends on accurate diagnostics. However, some scenarios, particularly instances of thruster faults due to external factors, pose a hard diagnostic task. This is particularly challenging in the context of abnormal behaviours that are detected but no fault diagnosis can be provided by the onboard fault management system. This type of fault is called non-diagnosable and it is the main target of this work. The aim of this paper is to propose a solution for controlling UUVs subject to non-diagnosable thruster faults using a Deep Reinforcement Learning (DRL)-based approach. This paper provides a comparison between an end-to-end DRL-trained controller and a standard PID controller to overcome partial and total thruster faults of a UUV. The consistency and robustness of the proposed method is verified by simulations. The results demonstrate the DRL-based controller's effectiveness in addressing non-diagnosable thruster faults that would otherwise hinder the successful completion of the mission.**

***Index Terms*— Unmanned Underwater Vehicle, partial thruster faults, non-diagnosable faults, thruster fault recovery, Deep Reinforcement Learning**

I. Introduction

Unmanned Underwater Vehicles (UUVs) are increasingly used in both civil and military applications, such as underwater exploration, mapping of the seabed, oceanographic data collection, wreck search and mine counter-measurement. However, during a mission, UUVs are confronted with faults due to the potential fragility and obsolescence of the system and the hostile environment within which the UUV is embedded. One of the most common types of UUV faults is thruster malfunction due to external factors such as collisions. A thruster fault can be partial, which means that the faulty component remains functional to a lesser extent, or it can be a failure, which is a permanent shutdown of the component functionality. If these malfunctions are not handled, they can lead to loss of control, termination of mission, or loss of drones. Existing methods for thruster fault management can't be effective without an accurate diagnosis. In this work, a non-diagnosable fault refers to the type of system anomaly in which, despite the detection of an abnormal behavior of the UUV, no fault diagnosis can be provided by the onboard sensors or analytical fault diagnosis methods. This occurs when the detected fault, in this case, a thruster fault due to external factors, does not match with any known faults of the system. The aim of this article is to focus on non-diagnosable UUV thruster faults and to propose a method based on DRL to control faulty UUVs, thereby facilitating a successful mission completion. This paper is organised as follows. Section II explores UUV dynamic modelling; then Section III covers existing thruster fault recovery methods and highlights their limitations and Section IV presents the innovative proposed solution based on UUV control with DRL [1]–[3]. The performance and effectiveness of our approach are evaluated in Section V through simulation results.

* This work is supported by French Defence Innovation Agency (AID)
[1] Lab-STICC UMR CNRS 6285, ENSTA Bretagne, Brest, France
[2] Naval Group Research, Ollioules, France
[3] Flinders University, Adelaide, Australia
[4] CROSSING IRL CNRS 2010, Adelaide, Australia

II. UUV dynamic modelling

RexROV2 (Fig. 1) is a hovering-type remotely operated vehicle designed for underwater exploration missions which require high manoeuvrability. RexROV2 possesses six degrees of freedom and is propelled by 6 thrusters, including 3 vertical thrusters, $T0$, $T4$ and $T5$, and 3 horizontal thrusters, $T1$, $T2$ and $T3$ as shown in Fig. 2.

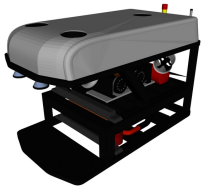

Fig. 1: RexROV2 design [4].

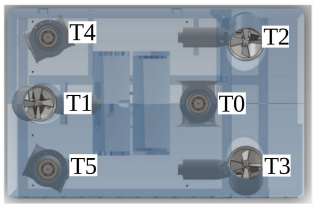

Fig. 2: RexROV2 thrusters configuration [5].

The complete modelling of the RexRov2 platform is challenging [5], [6], but it can be summarised in the state-space representation form [7] as:

$$\begin{aligned} \dot{\eta} &= J_{\Theta}(\eta)\nu \\ M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) &= \tau \end{aligned} \tag{1}$$

Where, in the first equation, η denotes the position and orientation vector with coordinates in the earth-fixed frame, ν denotes the linear and angular velocity vector with coordinates in the body-fixed frame and $J(\eta)$ is the kinematic transformation matrix. In the second equation, M is the inertia matrix, $C(\nu)$ is the matrix of hydrodynamic Coriolis and centripetal terms, $D(\nu)$ is the vector of hydrodynamic damping effects, $g(\eta)$ is the vector of gravity and buoyancy effect and τ is the *control input*, used to describe the forces $F_{x,y,z}$ and moments $M_{x,y,z}$ acting on the vehicle in the body-fixed frame:

$$\tau = \begin{bmatrix} F_x & F_y & F_z & M_x & M_y & M_z \end{bmatrix}^T \quad (2)$$

The forces are distributed among the thrusters as proposed in [7]:

$$\tau = Bu \in \mathbb{R}^n \quad (3)$$

Where $u = [u_1, u_2, ...u_r]^T \in \mathbb{R}^r$ is the vector of actuator commands, or *thruster input*, and B the Thruster Control Matrix (TCM) of size (n, r).

This UUV is also equipped with an Inertial Measurement Unit (IMU) that returns the velocity and orientation (in Euler angles). These variables are accessible through ROS topics [8]. Our software architecture consists of using the simulation meta-data to train the learning algorithms considered in this work.

III. UUV THRUSTER FAULT RECOVERY

As this study is centred on the topic of thruster fault management, the present section elucidates UUV thruster faults and reviews the methodologies presented in existing literature for mitigating these faults.

A. Origins and types of UUV thruster faults

Diverse sources can give rise to UUV thruster faults, rendering the task of pinpointing their origins challenging. For instance, during a mission described in [9], a UUV encountered a propeller blockage due to ice, which led to the servo-amplifier operating in current limitation mode. The ingress of saltwater during the same mission resulted in electrical dispersion, disrupting feedback signals and causing an increase in blade rotation speed. In [10], collisions with solid surfaces caused damage to the thruster blades, resulting in a reduction in efficiency. Cavitation, another challenge described in [11], triggers corrosion and erosion in propellers, progressively reducing their effectiveness. Due to these types of unexpected events, UUVs can experience a wide variety of thruster faults that are identified in the literature and grouped into different types [12], [13]: UUV thruster faults can be either failures, such as zeroing or constant output (Eq. 4), or partial, like constant biases (Eq. 5), or time-dependent biases (Eq. 6) altering thrust efficiency. In the equations below, T_f is the faulty thruster output, and T the desired thruster output. Due to this variety, a fault recovery strategy for UUV thruster faults has to be set up. The next section details some existing strategies.

$$T_f = a, \quad \text{with } a \in \mathbb{R} \quad (4)$$

$$T_f = \alpha T + \beta, \quad \text{with } \alpha, \beta \in \mathbb{R} \quad (5)$$

$$T_f = \alpha(t) T + \beta(t) \quad (6)$$

B. Fault recovery strategy

In most cases, fault recovery involves a series of four steps [14]: *fault detection*, *fault isolation*, *fault identification* and *fault control*. Fault isolation and fault identification can be grouped as *fault diagnosis*.

The first step is fault detection, which is a binary indicator that recognises the presence of faults. The next step is fault isolation, which consists of determining the location of the fault within the system's components. The subsequent step is fault identification, which is to describe the fault. Finally, the last step is fault control, also known as fault-tolerance or fault-tolerant control. This step aims to modify the system's control mechanisms to overcome the fault, on the basis of fault detection and diagnosis. Effective methods for UUV thruster fault detection, based on model-based observers, can be found in [15]. Efficient methods for UUV thruster fault control such as thruster reallocation or trajectory redesign have also been explored in the literature, relying on fault diagnosis [12], [16]. However, thruster fault diagnosis can be challenging, impeding effective fault control. The next subsection provides an overview of UUV thruster fault diagnosis to better understand the limits of existing methods.

C. UUV thruster fault diagnosis

Analytical redundancy is the most common method used to diagnose UUV thruster faults. It aims to compare the inputs and/or outputs of components with prior knowledge of the behaviour of these components. The result of this comparison is called *residual*. The characteristics of the residual are then used to detect and diagnose faults. The subcategories of analytical redundancy methods consist of *Model-based*, *Signal-based*, and *Data-driven* approaches.

- Model-based methods rely on precise models or models that closely approximate the system. Model-based methods are widely used to diagnose thruster faults, and can be categorised into three distinct groups: *Observer-based methods* as in [17], [18], *Parameter estimation methods* as in [19], and *Parity space methods* as in [20].
- Signal-based methods involve analysing output signals obtained from the system to identify abnormal behaviour or deviations from expected patterns. This method relies on comparing measured signals with reference signals or predefined thresholds to detect the presence of faults. In [21], a wavelet transform method was used for thruster fault diagnosis.
- Data-driven methods refer to the use of data analysis techniques and algorithms to identify and analyse patterns, trends, and anomalies in the data collected from various systems or processes. For example, in [22], a Deep Neural Network (DNN) was trained to identify UUV thruster faults.

The downside of all these approaches is that they require pre-existing knowledge of faults, which may not be feasible in some situations. For example, damage to the blades or the entanglement of foreign objects in the thrusters prove to be unpredictable and challenging to model accurately. In the absence of effective diagnostics, current approaches fall short of overcoming these issues, jeopardising the UUV's mission. This creates an opportunity to investigate the application of DRL methods, as presented in the next section.

IV. DRL-BASED PROPOSED SOLUTION

DRL is a machine learning approach that combines reinforcement learning and deep neural networks [23]. A DRL

agent learns optimal decision-making by interacting with its environment. The agent takes actions a_t in environment at time t, receiving a new state s_{t+1} and a reward r_{t+1}. The goal is to maximise the expected cumulative reward $J(\pi)$:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right] \tag{7}$$

Where γ is the discount factor that balances future rewards. The agent aims to find the optimal policy π^* that maximises this expected return. Existing methods are based on modelling the problem as a Markov decision process, notably relying on the probabilistic framework and the Markov assumption that the future state transition and rewards depend only on the current state and action. DRL leverages deep neural networks to approximate the policy and value functions. The policy network $\pi_\theta(a|s)$ outputs the probability distribution of actions a given state s, while the value network $V(s)$ estimates the expected cumulative return from state s. The policy is optimised by gradient ascent, and the value function is updated through the Bellman equation:

$$V(s) = \max_a \left(\sum_{s'} P(s'|s,a) \left(r(s,a,s') + \gamma \mathbb{E}_\pi [V(s')|s'] \right) \right) \tag{8}$$

Here, $\max_a$ denotes the maximum over all possible actions. The equation expresses the value of a state s as the maximum expected sum of rewards when taking the best action a in state s and then following the optimal policy π. These concepts are used in this work to deal with non-diagnosable faults.

A. DRL-based method to overcome non-diagnosable thruster faults

To overcome non-diagnosable thruster faults, the proposed idea is to bypass the diagnosis step and proceed directly to the fault control stage. For this purpose, a DRL-based method is proposed. This approach is motivated by the need for a controller capable of adapting to undiagnosed faults, based exclusively on the drone's behaviour. A description of the strategy is given here.

During training, the agent learns to overcome non-diagnosable thruster faults, thus acquiring the ability to address them during its mission. To achieve this, the agent will consistently encounter non-diagnosable faults and will be rewarded for successfully completing the mission, while facing penalties for failing to accomplish the mission.

The proposed strategy unfolds as follows: during the mission, the drone operates under an optimal model-based controller and when a fault is detected, two scenarios (as shown in Fig. 3) arise:

- If the fault is diagnosable, conventional control methods can handle the situation in an optimal way.
- If diagnostic methods onboard fail, a controller switch occurs, transitioning to the DRL-based controller previously trained on cases of undiagnosable faults.

To assess the efficacy of the proposed method, the focus is on scenarios where the fault has been detected yet remains non-diagnosable.

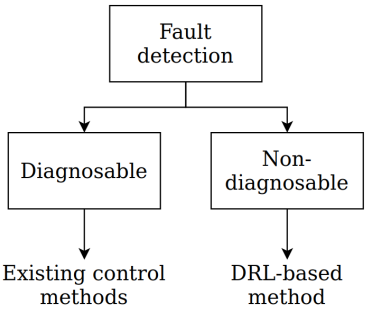

Fig. 3: Decision-making process when a fault occurs.

B. Environment setup

In this context, the chosen UUV's mission involves reaching a designated waypoint. For the evaluation of the proposed method, the drone's initial position is set at coordinates (0, 0, -9) and the target waypoint is positioned at (1, 1, -10). Non-diagnosable thruster faults, such as propeller damage or blocking, result in a reduction of thruster efficiency. This loss of efficiency can take various forms, but is considered in this paper for simplicity as a constant multiplicative bias in the thruster's output.

While all thrusters are susceptible to faults, the initial focus is directed towards the two thrusters with the most significant impact on the mission, as has been shown experimentally, namely $T0$ and $T5$ (Fig. 2). Empirical evidence indicates that when efficiency falls below 50%, i.e., when $\beta = 0$ and $\alpha = 0.5$ in Eq. 5, the drone faces challenges in reaching the designated waypoint. Hence, the idea is to train the agent on partial faults of these two thrusters, encompassing the cases outlined by Eq. 9:

$$\begin{aligned} T0_f &= \alpha T0, \quad 0 \leq \alpha \leq 0.5 \\ T5_f &= \alpha T5, \quad 0 \leq \alpha \leq 0.5 \end{aligned} \tag{9}$$

The agent is trained with a series of episodes, with a maximum number of steps denoted as nb_step_max. At each step, a partial thruster fault is applied as indicated in Algorithm 1. An episode ends either when the waypoint is reached or when the number of steps reaches nb_step_max.

Algorithm 1 Partial Thruster Fault Choice

1: **for** each episode **do**
2: Randomly choose a thruster, $T0$ or $T5$
3: Randomly choose α between 0 and 0.5
4: Apply partial thruster fault using the chosen thruster and α
5: **end for**

At each step, the agent chooses an action and receives a new state vector and a reward. As the UUV's goal is to reach the designated waypoint, the reward takes into account the position x, y and z of the drone. With the aim of maintaining a low rotation rate for the drone, a constraint is imposed on the rotation speed ($\dot{\theta}_x$, $\dot{\theta}_y$, $\dot{\theta}_z$) to be kept

close to zero. The UUV's rotation angles θ_x, θ_y and θ_z are not considered, enabling the agent to freely choose the appropriate orientation considering the faults. The UUV's linear speed is implicitly maximised within the reward, as a longer time taken to reach the waypoint results in a lower reward. The drone's linear speed is also not explicitly taken into account. As a result, the state parameters selected are the drone's position and its rotation speed. The state vector s is defined in Eq. 10. For simplicity, it is supposed that the position and rotation speed are measured accurately during the mission.

$$s = \begin{bmatrix} x & y & z & \dot{\theta}_x & \dot{\theta}_y & \dot{\theta}_z \end{bmatrix}^\intercal \tag{10}$$

The reward function for each step is described in Algorithm 2, with x_d, y_d and z_d the desired position, *i.e.* the waypoint position, and $\dot{\theta}_{xd}$, $\dot{\theta}_{yd}$, $\dot{\theta}_{zd}$ the desired rotation speeds, *i.e.* zero. The error limit below which the waypoint is considered to have been reached is denoted by ϵ.

Algorithm 2 Reward function

for each step **do**
  if nb_step $>$ nb_step_max **then**
    End of episode
  end if
  if waypoint reached, ie $e \leq \epsilon$ **then**
    $r = 100$
    End of episode
  else
    $e = |x_d - x| + |y_d - y| + |z_d - z| + |\dot{\theta}_{xd} - \dot{\theta}_x| + |\dot{\theta}_{yd} - \dot{\theta}_y| + |\dot{\theta}_{zd} - \dot{\theta}_z|$
    $r = -\exp(e)$
  end if
end for

The action vector of the agent is the control input τ described by Eq. 2, consisting of the forces and torques applied to the drone. Action values are continuous and bounded according to the values acceptable to the drone. To measure the performance of the proposed method, a performance index γ is introduced. This index is defined as the integral of the error e over an episode, which starts at $t = 0$ and ends at $t = t_{\text{end}}$:

$$\gamma = \int_0^{t_{\text{end}}} e(\tau)\, d\tau \tag{11}$$

The DRL-based controller chosen here is the Soft Actor-Critic (SAC) algorithm [24], because it is well-suited for underwater drone control due to its proficiency in continuous actions and adaptive exploration in unpredictable environments as shown in [2].

C. Soft Actor Critic algorithm

Soft Actor-Critic (SAC) has the following key features:

1) *Actor-Critic Architecture*: SAC uses two neural networks, an actor network to approximate the policy, and a critic network to estimate the state-action value function.
2) *Stochastic Policy*: the policy is modelled as a probability distribution over actions.
3) *Maximum Entropy Reinforcement Learning*: SAC maximises the expected reward while also maximising the entropy of the policy. This encourages the policy to explore more diverse actions and leads to robust learning.

The objective function of the SAC combines the expected reward and the entropy of the policy. The policy is updated by gradient ascent, and the critic is updated by minimising the Bellman error. The objective function of the SAC which is to be optimised is given by:

$$J(\theta) = \mathbb{E}_{(s,a)\sim D}\left[\alpha \log(\pi_\theta(a|s)) - Q_\phi(s,a)\right] \tag{12}$$

Where θ and ϕ are the parameters of the policy and critic networks respectively, D is the replay buffer, and α is the temperature parameter that controls the trade-off between exploration and exploitation.

V. Simulation Results

To assess the effectiveness of the proposed method, a comparison between a conventional PID-type algorithm and a DRL-based algorithm during a non-diagnosable faulty situation is conducted through simulations, and described in this section.

A. Simulation setup

The method proposed in this paper has been evaluated under simulations, by using a model of the RexROV2 drone included in the UUV Simulator package [4]. It is a ROS-Gazebo-based simulator, which includes packages needed for underwater robotics simulation. The evaluation of the two controllers in the various fault cases described in Eq. 9 is conducted on episodes where the fault is present throughout the entire episode. Each type of fault is evaluated over 100 episodes.

B. PID results

The PID controller is the most widely used control structure, and it is expressed as:

$$\tau = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)\, d\tau + K_d \cdot \frac{de(t)}{dt}, \tag{13}$$

where $e(t)$ is the output error at time t and K_p, K_i, K_d are respectively the proportional, integral, and derivative gains. The selected gains are the default gains of the UUV Simulator package. Here, the error vector e is defined as:

$$e = \begin{bmatrix} \tilde{x} & \tilde{y} & \tilde{z} & \tilde{\theta}_x & \tilde{\theta}_y & \tilde{\theta}_z \end{bmatrix}^\intercal \tag{14}$$

The results of the PID controller in the presence of partial faults of thrusters $T0$ and $T5$ are presented in Table I. This table shows the mean of the performance index over 100 episodes as a function of the considered faulty thruster and its efficiency rate. In green are the cases where the success rate is 100%, in orange where the success rate is between 60% and 70%, and in red where the success rate is 0%. It can

then be observed that, in the presence of fault, the controller mostly fails.

	Thruster $T0$	Thruster $T5$
Efficiency rate: 0.5	2.69 $\pm$0.47	2.54 $\pm$0.37
Efficiency rate: 0.4	3.26 $\pm$0.42	2.65 $\pm$0.43
Efficiency rate: 0.3	4.03 $\pm$0.25	2.83 $\pm$0.35
Efficiency rate: 0.2	4.30 $\pm$0.43	3.20 $\pm$0.46
Efficiency rate: 0.1	4.54 $\pm$0.44	3.38 $\pm$0.43
Efficiency rate: 0.0	4.82 $\pm$0.41	3.82 $\pm$0.48

TABLE I: Mean of the PID-method performance index γ over 100 episodes with the associated standard deviation.

C. SAC results

The agent was trained over 2400 episodes, each consisting of a maximum of 50 steps. The reward curve associated with the training is shown in Fig. 4, and the performance results during the evaluation are presented in Table II.

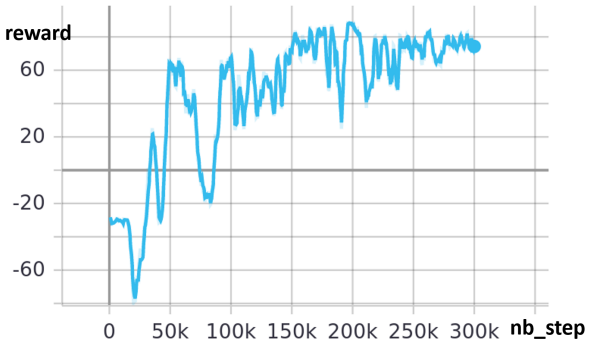

Fig. 4: Mean episodic training reward (averaged over 100 episodes) of the SAC method. The x-axis represents the number of steps, and the y-axis represents the reward value.

	Thruster T0	Thruster T5
Efficiency rate: 0.5	2.03 $\pm$0.31	1.75 $\pm$0.27
Efficiency rate: 0.4	2.15 $\pm$0.34	1.82 $\pm$0.36
Efficiency rate: 0.3	2.28 $\pm$0.35	1.87 $\pm$0.34
Efficiency rate: 0.2	2.56 $\pm$0.36	2.02 $\pm$0.32
Efficiency rate: 0.1	2.91 $\pm$0.43	2.18 $\pm$0.33
Efficiency rate: 0.0	4.12 $\pm$0.52	2.43 $\pm$0.33

TABLE II: Mean of the SAC-method performance index γ over 100 episodes with the associated standard deviation.

D. Results discussion

In order to evaluate the significance of the result, a method known as the Two-Sample Mean Comparison Test is used [25]. This method assesses whether two sample means differ significantly, considering their standard deviations and sample sizes. It is performed calculating a statistic test and comparing it to a critical value to determine significance (Algorithm 3).

Algorithm 3 Test Difference of Means

function TEST_MEANS($\mu_1, \sigma_1, n_1, \mu_2, \sigma_2, n_2, \alpha = 0.05$)
  $z_0 = \frac{(\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$
  $z_{\frac{\alpha}{2}} \cong 1.96$
  if $z_0 > z_{\frac{\alpha}{2}}$ or $z_0 < -z_{\frac{\alpha}{2}}$ **then**
    return True ▷ Significant difference
  else
    return False ▷ No significant difference
  end if
end function

This algorithm is applied to the results shown in Tables I and II to compare the PID controller and the DRL-based controller. It demonstrates that the DRL-based controller leads to a significant improvement for all the fault cases considered. Table III shows the result of $z_0 - z_{\frac{\alpha}{2}}$ for all fault cases.

	Thruster T0	Thruster T5
Efficiency rate: 0.5	9.76	15.29
Efficiency rate: 0.4	18.58	12.85
Efficiency rate: 0.3	38.72	17.70
Efficiency rate: 0.2	29.06	19.09
Efficiency rate: 0.1	24.52	20.18
Efficiency rate: 0.0	8.61	21.90

TABLE III: Calculation of the Two-Sample Mean Comparison Test $z_0 - z_{\frac{\alpha}{2}}$ for all fault cases, to compare the PID controller and the DRL-based controller. Positive values demonstrate the significance of the result.

To better understand the results, the UUV trajectories were presented in three different scenarios: the first involves no fault and is controlled by the PID (Fig. 5), the second simulates a fault in thruster $T5$, reducing its efficiency to 0.2, while still using the PID control (Fig. 6), and finally, the third scenario features the same thruster fault but uses the DRL-based control method (Fig. 7).

The Fig. 6 shows that when the UUV is under PID control and experiences the considered thruster fault, it fails to reach the designated waypoint: it can be observed that the drone is unable to descend sufficiently along the z-axis to reach the waypoint, and that the trajectory deviates along the x and y axis. This is attributed to the fact that thruster $T5$ is a vertical thruster, involved in both the movement along the z-axis and the rotation of the drone.

In contrast, the DRL-based method enables it to successfully navigate to the waypoint, as shown in Fig. 7, even if the trajectory does not follow a straight line. This can be interpreted by the fact that the DRL-based controller employs dynamic strategies to maintain control and compensate for the effects of the faulty thruster.

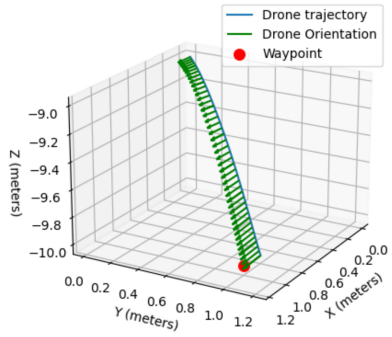

Fig. 5: UUV trajectory controlled by PID in the absence of fault.

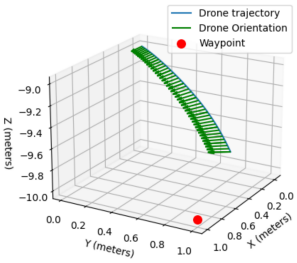

Fig. 6: UUV trajectory controlled by PID, where $T5$ efficiency rate is 0.2%.

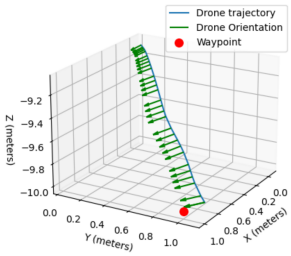

Fig. 7: UUV trajectory controlled by the DRL-based controller, where $T5$ efficiency rate is 0.2%.

VI. Conclusion

The development of safe and efficient autonomous vehicles is a key element in several applications of importance, such as exploration, inspection, delivery of goods and assisting people. However, the safe and effective control of these vehicles is hampered by the inability of traditional control systems to adapt to non-diagnosable faults. This problem is even more pronounced for underwater vehicles that need to operate in remote conditions. This paper presented a fault management strategy for non-diagnosable faults using a DRL-based controller. This strategy has been validated on two thruster faults for waypoint navigation with a strong improvement; based on a SAC algorithm, a DRL-based controller for UUV has been compared a classical PID controller. The evaluation of the results, through a performance index, shows the superiority of the DRL-based controller over the PID controller to overcome all the considered thruster faults. However, work is underway for further validation on more complex missions, including comparisons of different DRL algorithms and consideration of multiple simultaneous thruster faults.

References

[1] Y. Sola, T. Chaffre, G. Le Chenadec, K. Sammut, and B. Clement. Evaluation of a deep-reinforcement-learning-based controller for the control of an autonomous underwater vehicle. In *Global Oceans*, 2020.

[2] T. Chaffre, G. Le Chenadec, K. Sammut, E. Chauveau, and B. Clement. Direct adaptive pole-placement controller using deep reinforcement learning: Application to AUV control. *IFAC Conf. on Control Applications in Marine Systems (CAMS)*, 2021.

[3] T. Chaffre, P.E. Santos, G. Le Chenadec, E. Chauveau, K. Sammut, and B. Clement. Learning Adaptive Control of a UUV using A Bio-Inspired Experience Replay Mechanism. *IEEE Access*, 2023.

[4] M. Manhaes, S. Scherer, M. Voss, L. Douat, and T. Rauschenbach. UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation. In *MTS/IEEE OCEANS Conference*, 2016.

[5] V. Berg. *Development and Commissioning of a DP system for ROV SF 30k*. PhD thesis, NTNU, 2012.

[6] R. Yang, B. Clement, A. Mansour, M. Li, and N. Wu. Modeling of a complex-shaped underwater vehicle for robust control scheme. *Journal of Intelligent and Robotic Systems*, 2015.

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

[8] M. Quigley et al. ROS: an open-source Robot Operating System. In *Proc. of the ICRA Workshop on Open Source Robotics*, Japan, 2009.

[9] M. Caccia, R. Bono, G. Bruzzone, G. Bruzzone, E. Spirandelli, and G. Veruggio. Experiences on actuator fault detection, diagnosis and accomodation for rovs. *International Symposiyum of Unmanned Untethered Sub-mersible Technol*, 2001.

[10] W. Abed, S. K. Sharma, R. Sutton, and A. Khan. An unmanned marine vehicle thruster fault diagnosis scheme based on ofnda. *Journal of Marine Engineering & Technology*, 2017.

[11] C. Tsai, C. Wang, Y. Chung, Y. Sun, and J. Perng. Multi-sensor fault diagnosis of underwater thruster propeller based on deep learning. *Sensors*, 21, 2021.

[12] E. Omerdic and G. Roberts. Thruster fault diagnosis and accommodation for open-frame underwater vehicles. *Control Engineering Practice*, 2004. Guidance and control of underwater vehicles.

[13] Y. Sun, X. Ran, Y. Li, G. Zhang, and Y. Zhang. Thruster fault diagnosis method based on gaussian particle filter for autonomous underwater vehicles. *International Journal of Naval Architecture and Ocean Engineering*, 2016.

[14] M. Tipaldi and B. Bruenjes. Survey on fault detection, isolation, and recovery strategies in the space domain. *Journal of Aerospace Information Systems*, 12(2), 2015.

[15] A. Alessandri, M. Caccia, and G. Veruggio. Fault detection of actuator faults in unmanned underwater vehicles. *Control Engineering Practice*, 7(3), 1999.

[16] S. Ahmadzadeh, M. Leonetti, A. Carrera, M. Carreras, P. Kormushev, and D. Caldwell. Online discovery of auv control policies to overcome thruster failures. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[17] Zhenzhong C., Fei M., Daqi Z., and Chaomin L. Fault reconstruction using a terminal sliding mode observer for a class of second-order mimo uncertain nonlinear systems. *ISA Transactions*, 97, 2020.

[18] A. Baldini, A. Fasano, R. Felicetti, A. Freddi, S. Longhi, and A. Monteriù. A model-based active fault tolerant control scheme for a remotely operated vehicle. *IFAC-PapersOnLine*, 2018. Symposium on Fault Detection, Supervision and Safety for Technical Processes.

[19] L. Liu, W. Yu, and Z. Yu. Active fault-tolerant control design for a submarine semi-physical simulation system. *International Journal of Control, Automation and Systems*, 2018.

[20] A. Okada, A. Silva de Morais, L.C. Oliveira-Lopes, and L. Ribeiro. A survey on fault detection and diagnosis methods. In *14th IEEE International Conference on Industry Applications (INDUSCON)*, 2021.

[21] Y. Baoji, Y. Feng, W. Yujia, Z. Mingjun, and Z. Chenguang. Fault degree identification method for thruster of autonomous underwater vehicle using homomorphic membership function and low frequency trend prediction. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 2019.

[22] D. Stojcsics, D. Boursinos, N. Mahadevan, X. Koutsoukos, and G. Karsai. Fault-adaptive autonomy in systems with learning-enabled components. *Sensors*, 21, 2021.

[23] V. François-Lavet, P. Henderson, R. Islam, M. Bellemare, and J. Pineau. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 2018.

[24] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 2018.

[25] G. Snedecor and W. Cochran. *Statistical Methods*. Iowa state University Press, Ames, Iowa, 1989.