Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

A Soft Actor-Critic Approach to Complex Dynamic Flight Control for a Flying Firefighter Robot with Water-Jet Propulsion

Syafiqah Lagiman[1], Abu Ubaidah Shamsudin[1,*], Zubair Adil Soomro[1], Mohamad Hafiz Abu Bakar[1], Ruzairi Abdul Rahim[2], Carl John Salaan[3]

[1] Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, Batu Pahat Johor, Malaysia
[2] Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81300 Johor Bahru, Johor, Malaysia
[3] College of Engineering and Technology, Mindanao State University-Iligan Institute of Technology, Iligan City, Philippines

ARTICLE INFO

Article history:
Received 28 June 2023
Received in revised form 10 November 2023
Accepted 21 November 2023
Available online 8 December 2023

Keywords:
Deep Reinforcement Learning (DRL); Soft Actor Critic (SAC); flying fire fighter robot

ABSTRACT

Most of the robots nowadays become more intelligent in line AI technology are being applied in industry. Deep Reinforcement Learning (DRL) is involves gaining knowledge by making errors and fixing them. Hence, the robot will learn from mistakes and keep improving by process. Deep Reinforcement Learning is a kind of machine learning method that focuses on teaching models to act in ways that maximize rewards in an environment. Since the algorithm receives feedback in the form of rewards or consequences for its actions, this is often accomplished through trial and error. Soft Actor Critic (SAC) is the most recent Deep Reinforcement Learning technique. In this research, to design a simulation robot that can be trained by using Soft-Actor Critic. This purpose is to train and evaluate the performance of robot in MATLAB and VMware platform. According to this study, robots might learn and adapt to their surroundings more quickly and effectively with the aid of a system that utilises a suitable reward together with a guidance technique. Decisively, the simulation shows that the Soft Actor Critic algorithm successfully simulates the control of flying fire fighter robot in terms of stability.

1. Introduction

Fires are one of the most common calamities that occur in Malaysia every year, with almost 50,000 cases reported each year. Unfortunately, these issues are predicted to increase in the future, and there is an urgent need for more efficient and effective means of reducing the risk of fatality and injury. The unpredictable scale of these disasters is beyond the scope of human mission planning, and critical high-temperature situations and complex environments result in extremely hopeless conditions and further difficulties for rescue operations. The application of robotic technology can significantly improve disaster response, recovery, preparedness, and efficiency, while ensuring the safety of responders, and could also be the solution to these issues.

* *Corresponding author.*
E-mail address: ubaidah@uthm.edu.my

https://doi.org/10.37934/araset.34.2.271286

However, current robots cannot exhibit the same level of performance in the extreme environment of disasters as they can in controlled indoor or laboratory environments. Their ability to respond to unexpected situations is low, and there is a need for robots that can adapt to changing conditions and perform critical tasks under high-stress conditions [1]. In this work, the focus is on creating robots that can be used in unknown extreme environments where the situation is always changing, such as sudden fire explosions, falling building structures, death, and major injury [2]. When fighting a fire, for example, it is preferable for firefighters to be stationed far from the flames and to directly spray water on the fire source. However, direct access to the fire source is challenging, and it may be difficult for a firefighter to extinguish a fire that has spread throughout a building, especially if it is a large-scale blaze. The goal in this situation is to keep the fire inside the building from spreading outdoors.

To address this challenge, a number of traditional robot systems have been developed, including fire hose-equipped terrestrial mobility robots or search robots [3-5]. However, these robots are often built to walk on flat surfaces, and accessing the interior of a burning building can be difficult. These robots are categorized by Amano [6], who believes that aspects to think about include size and weight. Various research has suggested a solution to forest or metropolitan fires, where many flying robots fly in formation to eliminate the flames [7,8]. Ghamry *et al.,* [7] created an algorithm to determine an unmanned aerial vehicle's (UAV) best flight formation for spotting forest fires. Yamada *et al.,* [8] introduced the idea of remotely extinguishing a flame by mounting a drone to the end of a fire hose. However, this research has not yet been applied practically, such as by performing an action like fire extinguishing. To move the hose of the UAV closer to the fire source, issues like the payload and the impact of the hose's tension on the flight must be overcome.

Hence, in this study, a novel flying robot with water-jet propulsion for a flying firefighter robot is proposed. The concept of the flying fire-fighter robot using water jet propulsion for extinguishing fire on high-rise buildings is introduced. In robotics, flying robots with water-jet propulsion have emerged as one of the most promising robotic approaches in solving the limited mobility of conventional robots with large rigid structures and heavy payloads. The robots have higher mobility to reach high places by flying, and soft-robotic; slim design that can move in small opening areas and non-rigid designs that can change shape depending on their environment.

However, the robot's attachment to a non-rigid water hose causes complex mathematical modelling. Although several researchers have proposed different types of modelling recently, this study is different from such previous works in terms of the novel concept of the flying soft-robotic mechanism using high-pressured water propulsion. This study aims to control the complex flying water hose robot that can steadily fly. The robot head is a rigid body with nozzles, and the hose-like body is composed of a flexible hose. The study models soft-robotics using three assumptions. Firstly, the hose-like body of the soft-robot is divided into multiple links connected by elastic joints and dampers [9]. Each link is treated as a rigid body and each joint can bend in three different directions. Secondly, the model has its edge fixed at the origin and there is no ground. Lastly, the reaction force produced by the water jet is treated as an external force applied to the center of mass (COM) of each link, ignoring the fluid-structure interaction. The model parameters are based on the actual soft-robot and system identification is used to estimate the spring stiffness and damper coefficient for each joint. However, the complexity of the soft-robotics' dynamic model makes it difficult to develop a control system.

Boston Dynamics has created advanced robots with mobility, dexterity, and intelligence. The motion controllers for wheeled mobile robots are often based only on their kinematics. However, to reduce tracking errors, it's important to consider the dynamics of the robot, especially when performing high-speed movements or transporting heavy loads. Commercial mobile robots typically have internal controllers that accept velocity commands, but control signals generated by dynamic controllers are usually torques or voltages [10].

To address this issue, researchers have developed advanced control strategies that take into account both the kinematics and dynamics of the robot. For instance, the use of feedback control can be an effective way to control the robot's motion and prevent instability. Additionally, optimal control techniques can be used to determine the optimal motion trajectory of the robot, which can help to reduce tracking errors and improve the robot's overall performance.

Another key area of research in robotics is the development of machine learning algorithms for robot control. Reinforcement learning is a method of machine learning that focuses on how software agents should perform actions in an environment. It is a part of deep learning that allows us to optimize some aspect of the cumulative reward. Reinforcement learning allows robots to learn a wide range of performance skills. However, robotic reinforcement learning implementations often sacrifice the individuality of the learning process to reduce training times with actual systems. Deep Reinforcement Learning (DRL), which trains general-purpose neural network policies, alleviates this limitation. However, direct deep reinforcement learning algorithms have been limited to simple functions in simulated settings because of their high sample complexity.

Model-based deep reinforcement learning has shown promising ability to control complex robots such as flying soft-robotics. This type of control requires continuous control which is possible through deep reinforcement learning. The deep-learned control has shown robustness in real-world scenarios. In this case, a proposal is made to use deep reinforcement learning and the Boston Dynamics controller for a firefighter robot. DRL requires a model to learn the complex flight control of soft-robotics which allows for frequent model updates and is easier to train outside of policy [11]. Soft Actor-Critic (SAC) is the state-of-the-art DRL algorithm for use in real-world robots, which uses off-policy updates combined with a stable stochastic actor-critic formulation.

The project’s objective is to develop an SAC controller using MATLAB software to ensure the stability of the robot's flying movement and analyze its performance. The project scope includes controlling the robot's stability using dynamic controllers in MATLAB software with Gazebo simulator, enabling the robot to fly up to 1m high, and ensuring locomotion stability using Lagrange dynamic.

This paper presents the details of the proposed robotic technology, methodology, and expected outcomes. The paper is structured as follows: section II provides background information on firefighting robots and their limitations. Section III discusses the methodology used in this study, including the materials and instruments used. Section IV explains the expected outcomes of the study, including the implications for the field and future research. Finally, section V summarizes the main points of the introduction and previews the upcoming sections of the paper.

2. Literature Review

This chapter discusses previous studies related to the construction and formulation of a firefighter robot using dynamic control. Both the approaches and procedures for this research are clarified. To ensure that the project meets its aims, the literature study provided analyses and reports on industrial science related to this project.

Robots come in various forms, from walking on two, four, six or more legs, to flying in the air, to performing surgeries, and working in factories [12]. Model-based Deep Reinforce Learning Artificial Intelligence has shown a promising ability to control complex robot automation. This complex control requires continuous control that is possible by using deep- reinforce learning. The deep-learned control has shown robustness in a real situation, making it an ideal candidate for flying soft robotics. Formulating Deep Reinforce learning requires modelling for learning the complex flight control of soft robotics. There have been several firefighter robot experiments conducted previously, and the functionality, speed, and performance of these robots differ. The Tactical Hazardous Operations Robot (THOR) as shown in Figure 1 (left) is a humanoid robot designed for the Shipboard Autonomous Firefighting Robot (SAFFiR) program of the United States Navy [13,14]. It is a humanoid robot that can traverse dangerous floors on spacecraft as well as use hoses and a gateway. THOR can walk and run semi-autonomously with the assistance of a remote operator. However, the machine is inefficient and exposed to damage from fire and water. The Thermite RS1 as shown in Figure 1 (right) is a remote-controlled robot designed to traverse dangerous terrain and withstand exposure to extreme elements [15,16]. The turbine-aided firefighting robot (TAF 20) is a sophisticated robot firefighter that can clear blocking cars, expel smoke from a house, and blast water mist or foam [17]. When fighting major burning, this positions the firefighters at far less risk enabling TAF to enter and extinguish the fire without endangering anyone while reducing the risk of explosions [18]. The Fire Ox is an autonomous firefighting vehicle designed to combat fires and treat hazardous items as a first aid device [19].

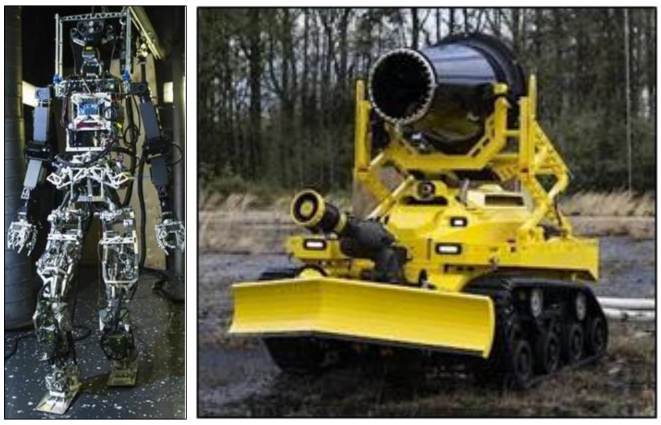
(a) (b)

Fig. 1. (a) Tactical Hazardous Operations Robot (THOR); (b) Thermite RS1 [13,15]

All the above robots are designed to improve on previous firefighter disadvantages. They are land-based, providing more stability on the ground, and can bring high workload, are heat resistant, remote controlled from a distance, and have many other functions. However, the disadvantage of these robots is that they cannot stay in water during combustion for too long as the momentum of the water is large. Water discharge is another weakness for the robot, as the pressure of water decreases as the distance between the fire hose and the burning center increases, causing the water not to reach the fire. Therefore, mist spray is the best solution to overcome burning. Nonetheless, the development of mist spray technology for firefighting robots is still in its early stages, and further research is required to improve its efficiency and effectiveness in extinguishing fires. In this research, the firefighter robot is upgraded as a flying robot by using a non-linear dynamic controller to improve the stability of the robot. The robot can be used in the higher building as this robot is using the 'drone concept'.

Dynamic control is a vital component in the design and construction of firefighter robots. Dynamic control involves the use of mathematical models and algorithms to control the movement and behavior of robots. In the case of firefighter robots, dynamic control is used to control the robot's movement, the orientation of the water nozzle, the amount and pressure of the water, and other important parameters.

In the journal of fire-fighter robot applications and controls, several articles discuss the problems faced by firefighters and the development of robots that could help them in extinguishing fires. In a study by Ando *et al.,* [20], the author discusses the difficulties faced by firefighters in directly accessing the fire source and the limited effectiveness of current large and heavy robots. They propose a new type of robot hose that can fly through a water jet into the fire source, allowing access to the interior of the burning building. Results showed that a 2 m long robot could fly stably in the air and its head direction could be controlled.

Addressing the challenge of locating the flame source and limited access to extinguish it, Gao *et al.,* [21] proposes a modular firefighting robot design that integrates proximity, vision, and IR sensors. The robot sprays water from a manipulator, and the software includes tracking, obstacle avoidance, flame detection, and motion algorithms. An experimental modular intelligent fire extinguishing robot was built and tested in simulation, with successful results. Various control algorithms, such as machine learning and neural networks, could improve efficiency and stability.

Article by Ando *et al.,* [22] emphasizes the impact of firefighting operations on firefighters, proposing a flexible tube at the outlet of a nozzle to control the trajectory of the water jet and conducting a mist spray from a close distance to validate the cooling effect of water. Results suggest that direct firefighting and analysis of the potential to suppress fire from a near distance to the fire source is practical, and mist spray from a close distance was effective for fire suppression.

Meanwhile, Baum *et al.,* [23] highlights the danger a structure on fire poses to firefighters and trapped individuals, proposing an audio recognition algorithm to classify firefighting sounds such as distress signals and structural collapse. The network layout requires minimal hardware functionality and achieves an average accuracy of 85.7%.

However, the weaknesses of using flame sensors were solved using deep learning to detect fire in real-time, with detailed outcomes relative to machine learning [24].

The literature review by Quillen *et al.,* [25] compares various deep reinforcement learning (DRL) algorithms for robotic grasping skills, evaluating each algorithm based on four parameters. Double Q-learning (DQL) improves the performance of the algorithm, and Deep RL can learn grasping diverse objects from raw pixels with a 90% success rate in a simulator.

Deep reinforcement learning algorithms for robotic manipulation with asynchronous off-policy updates, with the study showing that the algorithm can scale to complex 3D manipulation tasks [26]. The algorithm can learn deep bionetwork policies effectively, and RL can learn a range of 3D manipulation skills in simulation and a complex door opening skill on real robots.

Article by Tai *et al.,* [27] deals with virtual-to-real deep reinforcement learning for mapless navigation of mobile robots, proposing a learning-based mapless motion planner that takes sparse 10-dimensional range observations and goal locations to the mobile robot coordinate frame as inputs and outputs continuous steering commands. The trained planner can be directly applied to unseen virtual and real environments.

An end-to-end approach using deep reinforcement learning for mobile robot navigation in an unknown environment was proposed, using a dueling architecture based double deep Q network (D3QN) for navigation [28]. The study concludes that mobile robots can learn the environment knowledge gradually and navigate to the target destination autonomously with an RGB-D camera only.

The implementation of wireless sensor networks (WSNs) with mobile robotic sensor nodes for spatiotemporal control, proposes deep reinforcement learning tree (DRLT) that uses DRL to increase the efficiency of identifying the most insightful sampling sites [29]. The study concludes that DRLT achieved the best performance on both average estimation error and the estimation error variance, and the proposed method can instruct the robotic sensors to avoid unnecessary sampling positions.

In this study, we will be using a DRL-based control system to construct a firefighter robot. This system will allow the robot to learn from its experiences and make decisions based on the rewards it receives. We believe that this approach will result in a more effective and efficient firefighter robot.

3. Methodology
3.1 Project Planning

Developing a calculation and simulation of the hardware is the main part of this project. The next step is to combine the calculation of the Lagrange Equation and simulation for implementation. Also, MATLAB coding was uploaded into the designed flying firefighter robot which later was controlled using DRL in ROS software. This process was carried out to observe flying firefighter robot's motion, stability, and other results.

3.2 Hardware Design

The constructed design of flying firefighter robot model is depicted in Figure 2 where the legs were controlled by DC motors through linear motion in three axis while its root is fixed and cannot move. These consisted of four nozzles as high-impact force makers on a solid surface to ensure floating stability. These nozzles were supported by a hose reel. It was supported by using an automatic ladder by a fireman truck which is controlled by a remote controller as shown in Figure 2. This will lower the lifetime risk to firemen by keeping working up to 15 high levels of buildings.

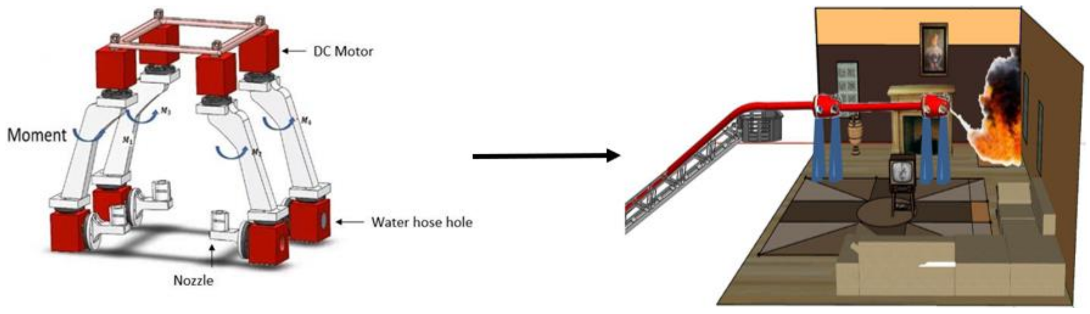

Fig. 2. Application of developed flying fire fighter robot in burning building

The proposed configuration of this flying fire fighter robot system is different from prior studies with a similar concept as this concept focuses primarily on a water-based propulsion system with a control valve that can provide sufficient actuation force for both translational and rotational motion without considering the forces and moments that cause it [22,30]. The exact structure of the additional propulsion actuator with a wire-based guiding mechanism will be described in a subsequent study.

The reference coordinate of the 6-DoF firefighting system is illustrated in Figure 3. The location of the body-fixed frame (b-frame, $o_b x_b y_b z_b$)) in the earth fixed frame (e-frame, oxyz) is defined by its position (x, y, z). The model is based on three assumptions: (A1) The hose type robot is approximated as having multiple links, which are connected by elastic joints and dampers. Each link is modelled as a rigid body and each joint can bend along three axes [22]. (A2) The edge of the robot model is fixed on the origin, and there is no ground. (A3) The reaction force, W induced by the nozzle is modelled as an external force applied on the centre of mass (COM) of link. We ignored the force caused by the fluid structure.

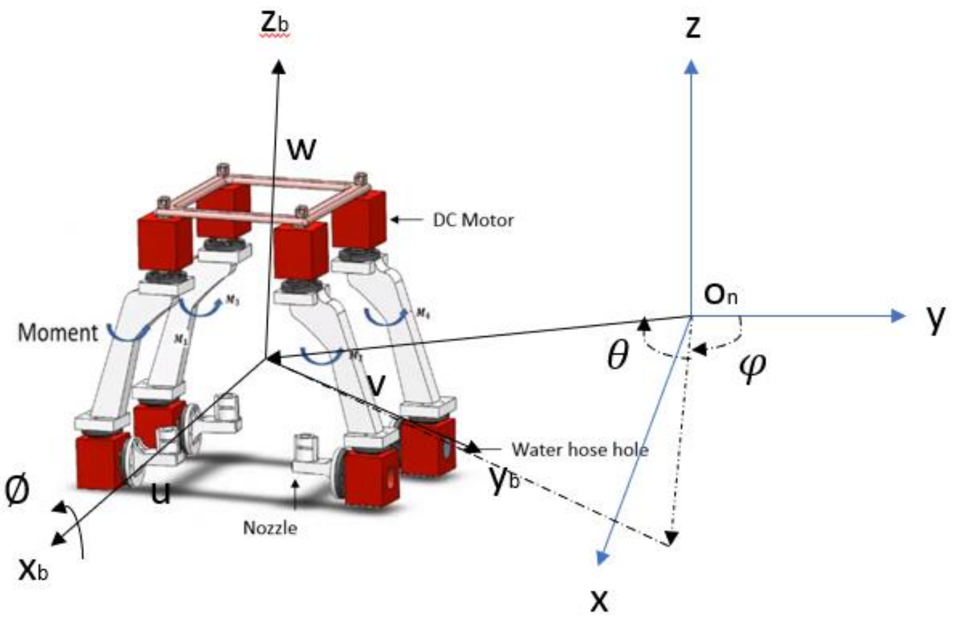

Fig. 3. Reference-coordinated description of flying firefighting system with water jet actuator

To ensure stable flying of the robot, we utilized the Deep Reinforcement Learning (DRL) control method. The net force f_i from the robot *i*(*i* = 1, 2, …) is controlled as follows:

$$f_i = F_i - D_{di}\dot{r}_i$$

The first term F_i is a constant force vector in the inertial coordinate system and determines the robot’s shape (equilibrium point). The second term is a damping term for the velocity of the nozzle, which suppresses vibrations and contributes to the convergence speed. In this experiment, we aim to achieve the net force as zero ($f_i = 0$) with constant water flow.

3.3 Deep Reinforcement Learning (DRL)

The process of controlling flying firefighter robot is shown in Figure 4 where DRL model containing policy, and the value network was jointly represented by a Network system. This technique was used for teaching the robot to learn emphasizing rewarding good conduct and/or forbids undesirable. It typically sensed and comprehended its surroundings, acted, and learned by mistakes [31]. This was chosen due to its ability to perform in complex conditions.

In this case, the experiments in the virtual world demonstrated the effectiveness of autonomous learning from the environment without any supervision signal. Particularly, Model Free Based type of Reinforcement Learning was used because the agent uses experience to learn Soft Actor Critic (SAC) policy or value function directly without using a model.

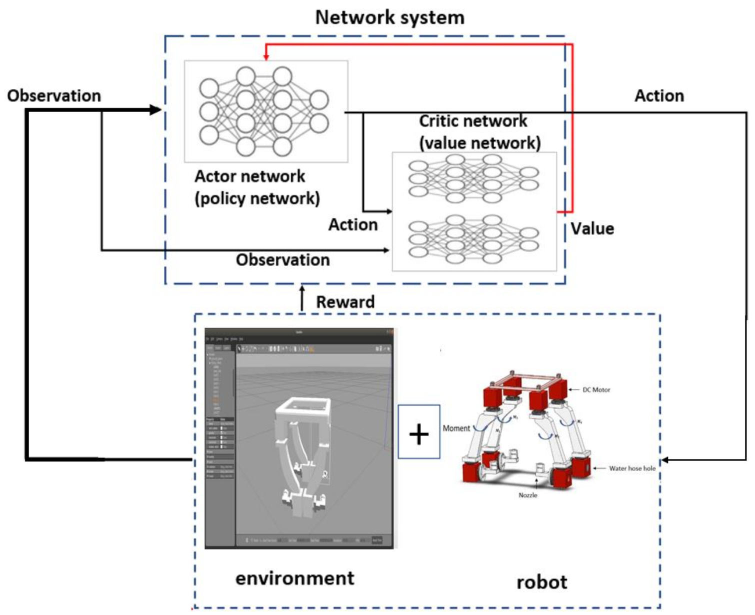

Fig. 4. An overall Deep Reinforcement Learning representation

3.3.1 Soft Actor Critic (SAC)

SAC is an off policy-critical based on the maximum entropy reinforcement learning paradigm. The actor's objective in this framework was to maximize the expected reward while also increasing entropy. Q-learning methods have been used to develop previous DRL methods based on this framework. Off-policy updates were combined with a stable stochastic actor-critic formulation in SAC as it is one of the most effective DRL algorithms in real-world robots [32].

DRL was applied to train an agent in how to perform a task in an unfamiliar environment. The agent delivered actions to the environment and receives observations and rewards from it as seen in Figure 5. The reward measured the success of an action with respect to completing the task goal. The agent contained two components: a policy and a learning algorithm. The policy mapped the chosen actions based on environmental observations. Meanwhile, the policy parameters were continually updated based on actions, observations, and reward [33].

In addition to using a policy gradient, SAC agents also use two neural networks to approximate the next action: the actor and the critic. The actor network takes the current state as input and outputs a probability distribution over the possible actions. The critic network takes the current state and action as input and outputs an estimate of the expected return. To create the agent and train the system to execute based on the actor and critic functions, the 'rlSACAgent' function was used [34]. This function learns a policy that maximizes the expected return. The policy is updated using a policy gradient method, and the value functions are updated using Q-learning. The rlSACAgent function has several options that can be used to control the learning process. These options include the learning rate, the discount factor, and the number of steps to take between policy updates. The SAC algorithm uses two loss functions, a policy loss and a value loss. The policy loss is used to update the actor network, while the value loss is used to update the critic network. The policy loss is calculated as shown in Eq. (1).

$$loss_{policy} = -E[Q(s,a) \times \log(\pi(a|s)] \qquad (1)$$

where $\pi(a|s)$ is the probability of taking action a in state s, $Q(s,a)$ is the value of state-action pair (s, a), and E[] is the expectation operator.

The value loss is calculated as shown in Eq. (2).

$$loss_{value} = (Q(s,a) - V(s))^2 \tag{2}$$

where $Q(s,a)$ is the value of state-action pair (s, a), and V(s) is the predicted value of state s.

Figure 5 shows a RL Agent simulated using a Simulink model as a training and simulation for environment.

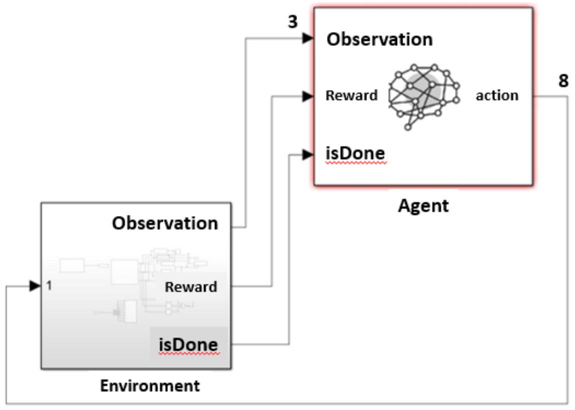

Fig. 5. A high-level view of the agent-environment interaction during training

3.3.2 Network systems

According to Figure 6, the output of the network is generated by a Gaussian distribution, which means that the data is generated randomly. To construct the Gaussian distribution layer, a combination of the mean and the standard deviation is used to determine the value of the output.

The actor network is made up of three layers, with 400, 300, and 8 neurons in each layer, respectively. The first layer takes in the current state of the environment as input, and the third layer outputs a vector of 8 values that represent the mean and standard deviation of a Gaussian distribution. The actor then selects an action by sampling from this distribution.

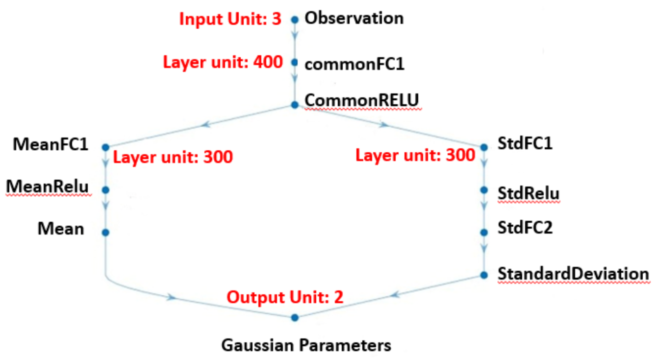

Fig. 6. Flow chart of actor network by using Deep network designer

The full connector layer is used in neural networks instead of the convolution layer because it can use all the features from the previous layer, rather than just a subset of them. This makes it more powerful and flexible, but also more computationally expensive. The ReLU activation function is used in almost the entire network because it helps prevent all the neurons from activating simultaneously, which can lead to overfitting. It also takes less time to train the network with ReLU activation than with other activation functions.

The critic network has a similar structure for both the critic 1 and critic 2 networks as shown in Figure 7. It is made up of two layers, with 400 and 300 neurons in each layer, respectively. The first layer takes in the current state of the environment as input, and the second layer outputs a value that represents the critic's estimate of the expected return from taking the current action in the current state. The critic network receives input from the observation and action, and the output of the critic network is used to update the Q-value for the actor network. The critic network only uses the feature input layer as the input part, the full connected layer as a connecting layer, and the ReLU activation function layer as the hidden layer for the network.

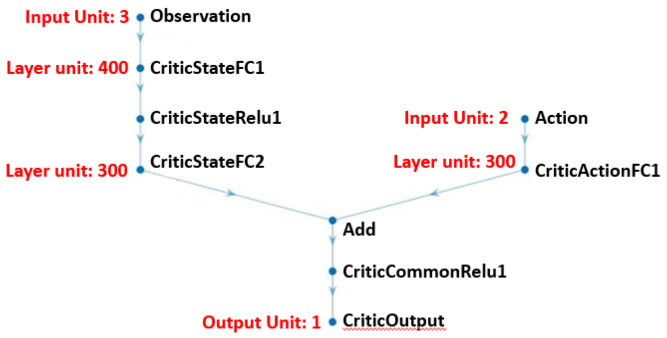

Fig. 7. Flowchart of 1st or 2nd critic network by using Deep network designer

In policy gradient reinforcement learning, the actor network is updated to increase the probability of taking actions that lead to high-quality state-action pairs. The critic network is updated to improve its estimate of the expected return from taking actions in different states.

3.3.3 Observation

The agent uses observations to improve its navigation control instructions for the robot as depicted in Figure 8. The robot uses three observation signals: position x-axis, y-axis, and z-axis. These signals are used to create an observation feature vector. The observation feature vector is then used to train the agent. The outputs of the agent were rotation of four motors in flying firefighter robot where SAC controller’s simulation was generated in Simulink.

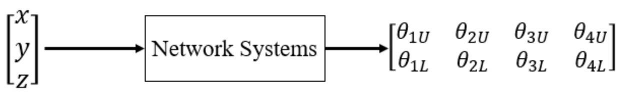
$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$
Network Systems
$\begin{bmatrix} \theta_{1U} & \theta_{2U} & \theta_{3U} & \theta_{4U} \\ \theta_{1L} & \theta_{2L} & \theta_{3L} & \theta_{4L} \end{bmatrix}$

Fig. 8. Block diagram of DRL agent input for observation and output

The agent determined the rotation of the motor by considering the robot's current position and target position. The value was updated as an observation parameter based on the linear velocity.

In this development, the action used to control the linear velocity of the movement is denoted by a_t. This study uses continuous actions to control the linear velocity of the movement. The action value in this project ranges between -1 and 1, as shown in Eq. (3).

$$a_t = \{\, linear\ velocity - 1 \leq a_t \leq 1 \tag{3}$$

The reward function uses the x, y, and z axis position to update the reward function based on the equation below:

$$\text{Reward} = (X = x*1) + (Y = y*1) + [(Z = z*1)\ [\text{if}(u1 < 1)]]+[(Z = z*1)/\ \text{elseif}(u1 > 1)] \tag{4}$$

The isDone function determines if the robot has reached its maximum boundary in all three axes. If the robot has reached its maximum boundary in any axis, the training process is terminated. The isDone function uses a logic gate to determine if the robot has reached its maximum boundary as shown in Table 1.

Table 1
Type of isDone using in this development

Type of IsDone	Condition
Position	*(X = x*1) < -1 = False* NOR *(X = x*1) > -1 = False* NOR *-1<(X = x*1) >1 = True*

where x and X is the current axis-x, axis-y, and axis-z and NOR is the NOR logic gate.

3.3.4 Training and simulation setup

For the simulation, each movement of the agent was recorded to validate the reward function's stability, acceleration, locomotion point, and others. To perform the experiment, the Gazebo was used to select all the 8 joints of the flying firefighter robot.

Furthermore, we analysed the collected data and evaluated the result based on the agent's ability to maintain stability from the initial to the target position.

4. Result and Discussion

The simulation results were used to evaluate whether the system developed meets the study's objectives or targets.

4.1 Training

In this study, the training of the agent was determined by two factors: the reward it received and the number of steps it took to complete a task. The results of the study, shown in Figure 9, indicate that the agent's performance improved over time for 5000 Episode reward. Early in the learning process, the agent was more likely to explore the environment, which resulted in fluctuating rewards and lower performance. This is likely because the agent was not yet familiar with the environment and did not know how to navigate it efficiently. As the agent learned more about the environment, it was able to take more efficient paths and achieve higher rewards. After 3.5 seconds, the agent had learned to navigate the environment effectively and was able to achieve consistently high rewards. However, it is important to note that the agent's performance was not perfect. The x, y, and z position functions were not as mature as the other functions, which suggests that the agent could still improve its performance in these areas. Overall, the results of the study indicate that the agent was able to learn to interact with the environment effectively and achieve high rewards.

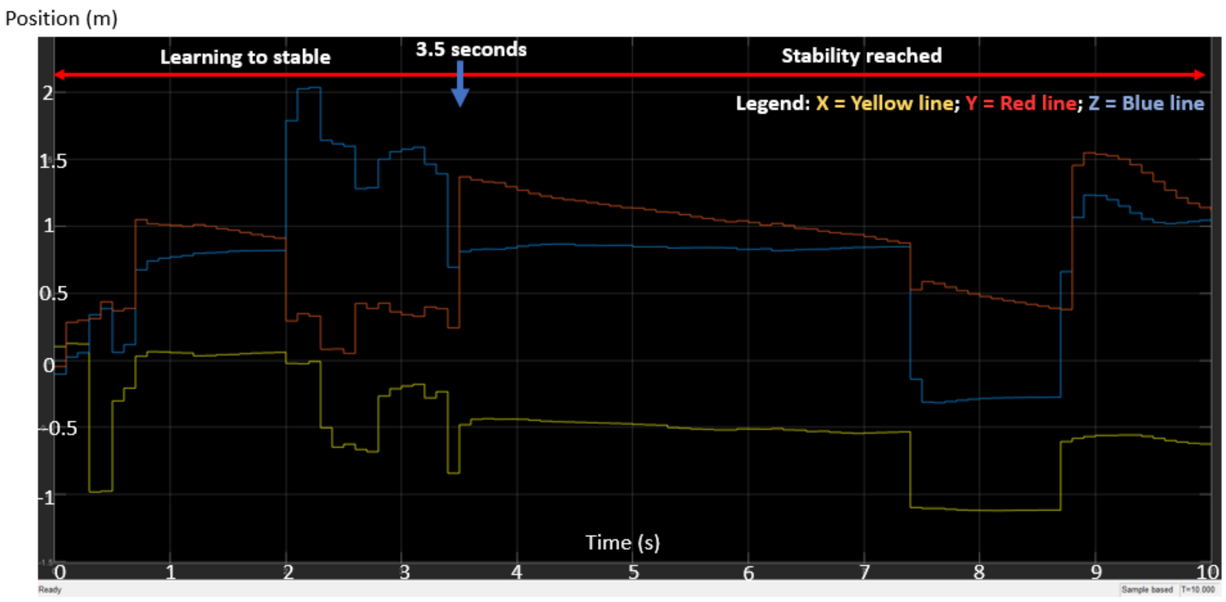

Fig. 9. Position X, Y and Z axis of robot for 5000 Episode reward

The trend of graph in Figure 10 shows the position of X, Y and Z axis of robot for 10000 Episode reward. Although trend of graph keeps fluctuating, it resulted in stability because the robot was able to manage its stability at a constant position.

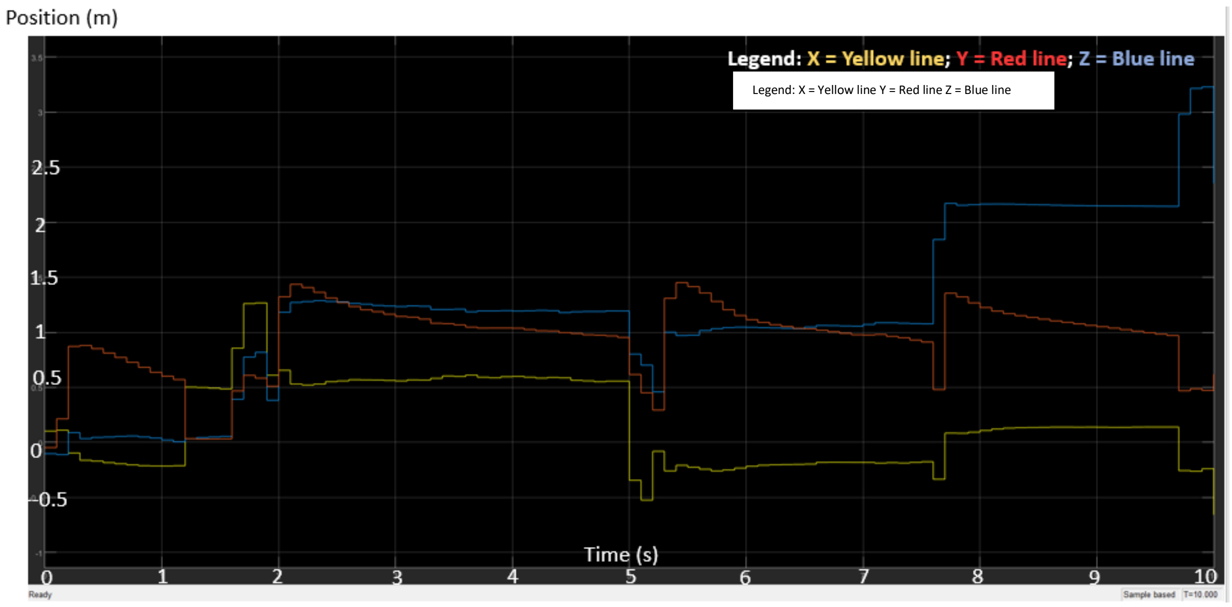

Fig. 10. Position X, Y and Z axis of robot for 10000 Episode reward

4.2 Simulation

Based on the data shown in Figure 11 and Table 2, the results showed the pattern of Time sampling versus Reward. The agent's movement to the goal location affects the simulation's outcome. From the graph, time sampling from 0 s to 2.2 s showed the increment of reward. This is because robot was in reinforcement learning process. From 2.2 s until 8.6 s, there was constant increasing until the peak reward, 6.5. In this state, robot is already adapted to new reinforcement learning environment. After 8.6 s until 10.0 s, there has been a decrement in graph due to robot already achieving maximum reward.

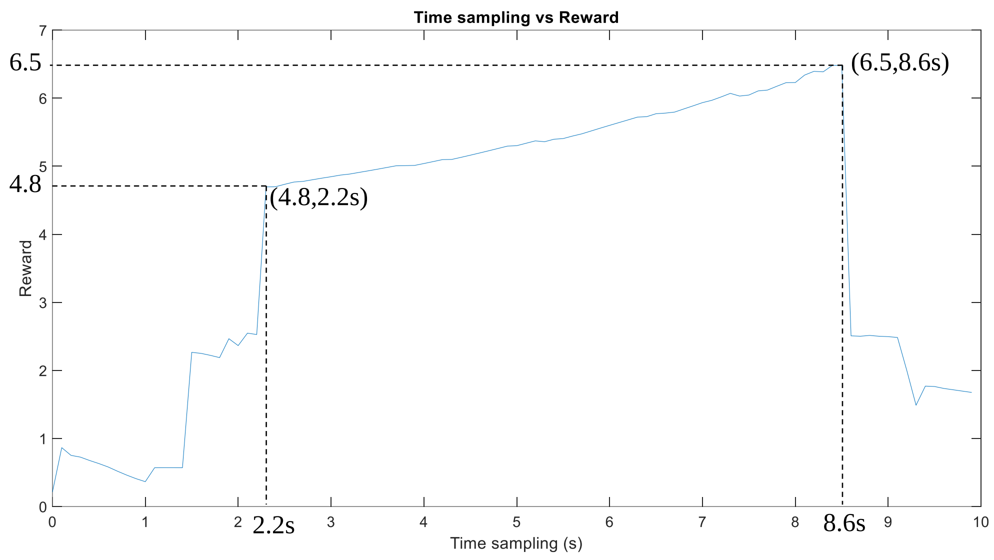

Fig. 11. Graph of Time sampling versus Reward for 10000 Episode Training

Table 2
Comparison time sampling between peak Reward for increment and decrement Time sampling

Time sampling (s)	Reward
2.2s	4.8
8.6s	6.5

The robot learned to navigate from an initial to a target location through reinforcement learning. As shown in Figure 12, the robot was able to achieve a stable state, which indicates that it had learned to navigate the environment effectively. The robot's accuracy in reaching the target position suggests that the reward system was effective in guiding the robot's learning. This experiment demonstrates that SAC training and simulation systems that uses continuous reward strategy can help flying robot learn more efficiently and reach stability.

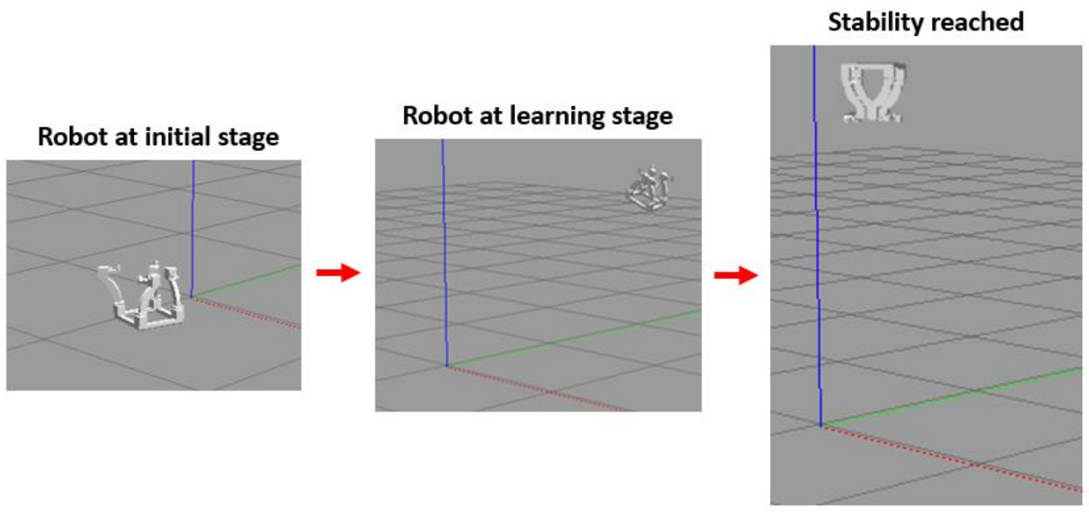

Fig. 12. Robot is in stable mode

The robot was able to learn how to reach a target position through reinforcement learning. The reward system was based on a continuous-state dynamic system, where the robot received a reward of more than 1 for reaching the target position and a reward of less than 1 for not reaching the target position. The results of the experiment showed that DRL is beneficial in the stability process, especially when using SAC.

5. Conclusion

In conclusion, this thesis has investigated the use of SAC Deep Reinforcement Learning for navigation with an optimal reward function. The results of the simulation showed that the robot was able to achieve stability after 10000 Episode rewards. The dynamic controller was developed using MATLAB software, and the SAC algorithm was able to compute an ideal strategy that optimized the entropy of the strategy while maximizing the long-term expected reward. The simulation results also showed that the robot was able to maintain stability from any pressure.

The research discussed in this thesis focuses on developing a dependable and consistent system that can be utilized as a robot controller using the SAC Reinforcement Learning concept with the reward function as the primary aspect that will affect the robot's performance.

For future work, it is advised to design for real-world use and to take the environment—particularly environmental impediments and weather—into consideration because these elements will have an impact on performance. It is also advised to liaise with fireman’s tools to make the robot more convenient with fireman’s needs. Additionally, it is advised to add more sensors to make it more effective for robots. Finally, it is advised to add a support stand to the robot while using in fire extinguishing operation.

Overall, the results of this research are promising and suggest that SAC Deep Reinforcement Learning is a viable approach for developing robots that can stable itself in complex environments.

Acknowledgements
This research is supported by Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme (FRGS/1/2022/ICT02/UTHM/03/1) and Universiti Tun Hussein Onn Malaysia (UTHM) through Contract Grant (Vot H381).

References

[1] Chen, Jerry, Maysam Abbod, and Jiann-Shing Shieh. "Integrations between autonomous systems and modern computing techniques: a mini review." *Sensors* 19, no. 18 (2019): 3897. https://doi.org/10.3390/s19183897
[2] Nilsson, Martin, Håkan Frantzich, and Patrick Van Hees. "Selection and evaluation of fire related scenarios in multifunctional buildings considering antagonistic attacks." *Fire Science Reviews* 2 (2013): 1-20. https://doi.org/10.1186/2193-0414-2-3
[3] Zhang, Weichun, and Cheng Dai. "Development of a new remote controlled emergency-handling and fire-fighting robot." In *2009 WRI World Congress on Computer Science and Information Engineering*, vol. 5, pp. 239-243. IEEE, 2009. https://doi.org/10.1109/CSIE.2009.473
[4] Zhang, Qin, and Guangzhen Ke. "Kinematic analysis of fire-fighting robot under the impact of waterflow recoil force." In *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 264-268. IEEE, 2015. https://doi.org/10.1109/ICMA.2015.7237494
[5] Rakib, Tawfiqur, and MA Rashid Sarkar. "Design and fabrication of an autonomous fire fighting robot with multisensor fire detection using PID controller." In *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pp. 909-914. IEEE, 2016. https://doi.org/10.1109/ICIEV.2016.7760132
[6] Amano, Hisanori. "Present status and problems of fire fighting robots." In *Proceedings of the 41st SICE Annual Conference*. SICE 2002., vol. 2, pp. 880-885. IEEE, 2002.
[7] Ghamry, Khaled A., Mohamed A. Kamel, and Youmin Zhang. "Multiple UAVs in forest fire fighting mission using particle swarm optimization." In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1404-1409. IEEE, 2017. https://doi.org/10.1109/ICUAS.2017.7991527
[8] Hwangbo, Jemin, Inkyu Sa, Roland Siegwart, and Marco Hutter. "Control of a quadrotor with reinforcement learning." *IEEE Robotics and Automation Letters* 2, no. 4 (2017): 2096-2103. https://doi.org/10.1109/LRA.2017.2720851

[9] Liljeback, Pål, Kristin Y. Pettersen, Øyvind Stavdahl, and Jan Tommy Gravdahl. "Controllability and stability analysis of planar snake robot locomotion." *IEEE Transactions on Automatic Control* 56, no. 6 (2010): 1365-1380. https://doi.org/10.1109/TAC.2010.2088830

[10] Martins, Felipe Nascimento, and Alexandre Santos Brandão. "Motion Control and Velocity-Based Dynamic Compensation for Mobile Robots." In *Applications of Mobile Robots*. IntechOpen, 2018. https://doi.org/10.5772/intechopen.79397

[11] Asadi, Kavosh. "Strengths, weaknesses, and combinations of model-based and model-free reinforcement learning." *Department of Computing Science University of Alberta* (2015).

[12] Guizzo, Erico. " What Is a Robot?." *Robots Guide*. August 2, 2018. https://robotsguide.com/learn/what-is-a-robot/.

[13] Hodson, Hal. " Robot firefighter puts out its first blaze." *NewScientist*. February 5, 2015. https://www.newscientist.com/article/dn26920-robot-firefighter-puts-out-its-first-blaze/.

[14] Bogue, Robert. "The role of robots in firefighting." *Industrial Robot: The International Journal of Robotics Research and Application* 48, no. 2 (2021): 174-178. https://doi.org/10.1108/IR-10-2020-0222

[15] Atadero, Julybien. " Firefighting Robot Ready For Duty: LA City Fire Dept. Acquires Howe & Howe Thermite RS3." *Hotcars*. October 14, 2020. https://www.hotcars.com/firefighting-robot-ready-for-duty-la-city-fire-dept-acquires-howe-howe-thermite-rs3/.

[16] Howe & Howe Technologies. "THERMITE®." *Howe & Howe Technologies*. Accessed July 20, 2023. https://www.howeandhowe.com/civil/thermite.

[17] Murphy, Mike. " Robots are now fighting fires in Australia." *Quartz*. December 11, 2015. https://qz.com/571881/robots-are-now-fighting-fires-in-australia.

[18] Highfield, Vaughn. " Meet TAF20, The Turbine-Aided Firefighting Robot Of The Future." *Alphr*. December 14, 2015. https://www.alphr.com/the-future/1002221/meet-taf20-the-turbine-aided-firefighting-robot-of-the-future/.

[19] Mike. "Fire Fighting Robots." *Fire Safety Tips*, 2023. https://firesafety.tips/fire-fighting-robots/.

[20] Ando, Hisato, Yuichi Ambe, Akihiro Ishii, Masashi Konyo, Kenjiro Tadakuma, Shigenao Maruyama, and Satoshi Tadokoro. "Aerial hose type robot by water jet for fire fighting." *IEEE Robotics and Automation Letters* 3, no. 2 (2018): 1128-1135. https://doi.org/10.1109/LRA.2018.2792701

[21] Gao, Shang, Zhiyang Zhang, Zihan Zhao, and Mohsin M. Jamali. "Vision and infra-red sensor based fire fighting robot." In *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 873-876. IEEE, 2018. https://doi.org/10.1109/MWSCAS.2018.8624080

[22] Ando, Hisato, Yuichi Ambe, Tomoka Yamaguchi, Masashi Konyo, Kenjiro Tadakuma, Shigenao Maruyama, and Satoshi Tadokoro. "Fire Fighting Tactics with Aerial Hose-type Robot "Dragon Firefighter"." In *2019 IEEE International Conference on Advanced Robotics and its Social Impacts (ARSO)*, pp. 291-297. IEEE, 2019. https://doi.org/10.1109/ARSO46408.2019.8948716

[23] Baum, Elizabeth, Mario Harper, Ryan Alicea, and Camilo Ordonez. "Sound identification for fire-fighting mobile robots." In *2018 Second IEEE International Conference on Robotic Computing (IRC)*, pp. 79-86. IEEE, 2018. https://doi.org/10.1109/IRC.2018.00020

[24] Ramasubramanian, Sreesruthi, Senthil Arumugam Muthukumaraswamy, and A. Sasikala. "Fire detection using artificial intelligence for fire-fighting robots." In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 180-185. IEEE, 2020. https://doi.org/10.1109/ICICCS48265.2020.9121017

[25] Quillen, Deirdre, Eric Jang, Ofir Nachum, Chelsea Finn, Julian Ibarz, and Sergey Levine. "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods." In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6284-6291. IEEE, 2018. https://doi.org/10.1109/ICRA.2018.8461039

[26] Gu, Shixiang, Ethan Holly, Timothy Lillicrap, and Sergey Levine. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates." In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3389-3396. IEEE, 2017. https://doi.org/10.1109/ICRA.2017.7989385

[27] Tai, Lei, Giuseppe Paolo, and Ming Liu. "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation." In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 31-36. IEEE, 2017. https://doi.org/10.1109/IROS.2017.8202134

[28] Ruan, Xiaogang, Dingqi Ren, Xiaoqing Zhu, and Jing Huang. "Mobile robot navigation based on deep reinforcement learning." In *2019 Chinese Control and Decision Conference (CCDC)*, pp. 6174-6178. IEEE, 2019. https://doi.org/10.1109/CCDC.2019.8832393

[29] Chen, Jiahong, Tongxin Shu, Teng Li, and Clarence W. de Silva. "Deep reinforced learning tree for spatiotemporal monitoring with mobile robotic wireless sensor networks." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50, no. 11 (2019): 4197-4211. https://doi.org/10.1109/TSMC.2019.2920390

[30] Lee, Dong-Hun, Thinh Huynh, Young-Bok Kim, and Chakir Soumayya. "Motion control system design for a flying-type firefighting system with water jet actuators." In *Actuators*, vol. 10, no. 10, p. 275. MDPI, 2021. https://doi.org/10.3390/act10100275

[31] Wang, David Yuchen, Harpriya Bagri, Calum Macdonald, Spencer Kiy, Paul Jung, Olivier Shelbaya, Thomas Planche, Wojciech Fedorko, Rick Baartman, and Oliver Kester. "Accelerator tuning with deep reinforcement learning." In *Workshop at the 35th Conference on Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada*. 2021.

[32] Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." In *International Conference on Machine Learning*, pp. 1861-1870. PMLR, 2018.

[33] Karunakaran, Dhanoop. "Soft Actor-Critic Reinforcement Learning algorithm." *Medium*. December 2, 2020. https://medium.com/intro-to-artificial-intelligence/soft-actor-critic-reinforcement-learning-algorithm-1934a2c3087f.

[34] MathWorks. "Soft Actor-Critic (SAC) Agents." *The MathWorks Inc*. Accessed February 15, 2023. https://www.mathworks.com/help/reinforcement-learning/ug/sac-agents.html.