Coordination, Communication and Robustness in Multi-Agents: An Industrial Network Scenario Using Trust Region Policy Optimization

Munam Ali Shah
Department of Computer Networks and Communication-College of Computer Science and Information Technology,
King Faisal University, Al-Ahsa, Kingdom of Saudi Arabia

***Abstract*—Numerous practical uses necessitate multi-agent systems, including managing traffic, assigning tasks, regulating ant colonies, and operating self-driving cars, and drones. These systems involve multiple agents working together, communicating and engaging with their surroundings to achieve the highest possible total numerical reward. Deep Reinforcement Learning (DRL) approaches are used to address these multi-agent applications. In many circumstances, the use of agents raise challenges to safety and robustness. To address these issues, we develop a DRL based system in which multiple agents in an industrial network scenario interact with the real-world environment and act collaboratively and cooperatively. In proposed model, several agents collaborate with one another to complete tasks and maintain a safe state. To take actions cooperatively and collaboratively of agents in accordance with the safety robustness of policies, we apply DRL algorithms such as proximal policy optimization (PPO) and Trust Region Policy Optimization (TRPO) algorithms and DRL approaches. We apply Curriculum Learning (CL) for their better performance and training. In this study, a reward structure is also proposed which help agents to maintain their safe state. Mean reward, policy loss, value loss, value estimate and safety robustness are analyzed as performance matrix in this study. The results shows that the policy adopted in the proposed model perform comparably better than the other policies.**

***Keywords*—Safety robustness; reinforcement learning; multi-agents; safe state; collaboration**

I. Introduction

In an experiment on a cat's behaviour done by Thorndike [1], reinforcement learning (RL) begins with the trial and error method. In machine learning, RL is a significant approach. It's a method of determining which action offers the most reward [2]. Real-time learning involves an agent that interacts with the learning environment, takes actions on the current state, and proceeds to the next state while striving to maximize the numerical reward. DRL, which merges Deep Learning and Reinforcement Learning [3] [4], is utilized to handle high-dimensional environments. To surpass the limitations of single-agent systems, multi-agent systems are necessary. The agents in multi-agent systems communicate with one another, interact with the environment, and aim to maximize the overall numerical reward [5]. Many real-world applications, such as traffic control, task distribution, ant colonies, autonomous cars, and drones, necessitate multi-agent systems. Due to the complexity of a high-dimensional environment, an agent that is hard-coded cannot work well in it.

To ensure the agents can generate optimal learning solutions, Multi-Agent Deep Reinforcement Learning (MADRL) is a crucial technique in the previously mentioned applications. In these applications, MADRL agents can discover the most effective solutions and policies. MADRL is a learning method where multiple agents interact with the learning environment to maximize the total reward. Using neural networks with numerous layers and approximations, agents strive to estimate the value of actions. Their policies rely on their own action values and those of other agents [5].

Therefore, researchers are now focusing not only on reward maximization but also on safety. Deep integration of machine learning and control technologies is required to tackle safety challenges in critical applications such as self-driving cars and robot manipulation [4]. In the control community, robust and adaptive control techniques have been developed to ensure the safety and stability of systems. To enhance the safety of learned policies, appropriate control approaches from the control community can be applied [4] [6]. The performance of learned policies is typically measured through success rates [7] [8] or obtained rewards [9] [10] to determine their robustness. However, when using RL to address problems in safety-critical applications, policy robustness must also be evaluated from a safety perspective. However, there is presently no way of evaluating policy robustness from the standpoint of safety.

This work proposes a framework for the safety robustness of agents using deep reinforcement learning techniques. In the proposed work, a system is developed in which multiple agents interact with the real-world environment and act collaboratively and cooperatively. This work is an extension to our previously published work [33]. Here, we have added policy loss, safety robustness, value loss and value estimate. We have evaluated these parameters on multiple policies, i.e., Policy 1, Policy 2 and Policy 3. We have also proposed a novel algorithm for the training process. In proposed model, several agents collaborate with one another to complete tasks and maintain a safe state. To take actions cooperatively and collaboratively of agents in accordance with the safety robustness of policies. We have applied DRL algorithms such as proximal policy optimization (PPO) and Trust Region Policy Optimization (TRPO) algorithms and DRL approaches.

The rest of the paper is organized as follows: Section II provides the literature review. Section III presents the proposed methodology. In Section IV, the details of the proposed model are presented. Section V explains the implementation setup. In Section VI, experiments and results are presented and the paper is concluded in Section VII.

II. Literature Review

A. Single Agent Scenarios

Alternative reward functions may be presented to help an agent learn better if a reward function for an RL problem is not provided. With multiple reward functions, the agent may learn a variety of policies with varying levels of robustness. Evaluating the achieved reward and success rate has been a common method for assessing the robustness of a policy. When utilizing RL to address issues in many safety-critical situations, safety is a concern. Although reward and success rate have been commonly used to evaluate policy robustness, there has been limited discussion on assessing policy robustness from a safety perspective. This study [11] contributes to the field by introducing a novel concept of safety robustness and an algorithm to estimate it. The proposed method is applied to evaluate the safety robustness of three policies for controlling the manipulation of a cable-driven parallel robot. The results demonstrate that the algorithm is effective in estimating safety robustness by comparing the number of safe episodes to the total number of episodes. The study also includes illustrative experiments to demonstrate the implementation of the proposed algorithm. It can also choose the safest policy from a set of many policies.

As a recent and popular research area, RL faces a number of issues. Agent navigation and obstacle avoidance are the most appealing of these RL challenges. It is well understood the value of precise and multidimensional monitoring in detecting problems early and preventing them from escalating. Previously, AI was employed for drone navigation, with images from on-board cameras being processed for wayfinding and collision avoidance. The authors of this paper presented a novel recommender system for drone navigation, which utilizes artificial intelligence and sensor data while requiring only basic information [12]. Their approach involves combining two techniques: Proximal Policy Optimization (PPO) for navigation with minimal information and Long-Short Term Memory (LSTM) networks for overcoming obstacles using navigation memory. The safety requirements of the system are determined using a systematic functional failure analysis (FFA).

RL methods have been used to handle a variety of sequential decision-making issues for a long time. In complex environments with high dimensions, algorithms face significant challenges when attempting to optimize policies. Recent advances in deep learning have enabled RL approaches to drive optimal policies for complex agents capable of performing well in these difficult contexts [13] [14]. In their paper, the authors conducted a survey on various approaches for multi-agent deep RL (MADRL). The survey covers non-stationary, partial observability, continuous state and action spaces, multi-agent training methods, and multi-agent transfer learning [15]. The survey also discusses the advantages, disadvantages, and applications of the evaluated approaches, leading to the creation of more robust and practical multi-agent learning methods for tackling real-world issues in the future.

To overcome challenges in dealing with local-stable-point scenarios in complex situations, the artificial potential field method needs to be revised, which may increase the algorithm's complexity. In this study, the authors proposed an improved black-hole potential field combined with RL [16] to address this issue. The black hole potential field is used as the environment in an RL system, where agents automatically adapt to their surroundings and learn how to discover targets using basic environmental data. Furthermore, with the curriculum learning technique, taught agents adapt to a variety of contexts. Meanwhile, the avoidance process is visualized to show how agents avoid obstacles and achieve their destination. Static and dynamic trials are used to assess our approach. The findings indicate that agents can autonomously learn how to escape local stability points even without prior knowledge.

The authors presented a collision avoidance system in their research paper, which aims to provide reliable autonomous navigation for self-driving ships. The system uses a collision risk assessment [17] technique based on the ship's domain and the closest point of approach (CPA) to estimate the collision risk, and generates an avoidance path only when necessary. The ship domain is designed with an asymmetric shape, taking into account the ship's maneuverability and the international regulations for preventing collisions at sea (COLREGs). The CPA is used to determine the value of a quantitative collision risk. The authors of [18] investigate the resilience of RL with adversarial altered state observations, which is relevant for deploying real-world RL agents in the face of unexpected sensing noise. They show that with a given agent strategy, an optimum adversary to tampering state observations can be discovered, ensuring that the worst-case agent reward is obtained. This results in a unique empirical adversarial assault on RL agents in DRL environments, via a learnt adversary that is significantly stronger than earlier ones. Authors offer an alternate training with learned adversaries (ATLA) approach to improve an agent's resilience by training an opponent online alongside the agent using policy gradients and the optimum adversarial attack framework.

The automobile industry is being encouraged toward more flexibility and stability in manufacturing by EU laws on CO2 limitations and the trend of individualization. Modular manufacturing, in which workstations are detached by autonomous guided vehicles and new control ideas are required, is one way to handle these issues. Throughput-optimal coordination of goods, workstations, and vehicles are the goals of modular production control. Conventional control techniques for this NP-hard issue are inefficient in computation, do not discover optimal solutions or are not generalizable. On the other hand, DRL provides strong and generalizable algorithms that can deal with a wide range of settings and high levels of complexity. The PPO method is utilized in solving modular production control, as discussed in the paper [19]. The experimental results showed that the learned behavior of the agent was optimal, reliable, consistent, and generalizable across various modular production control setups. The agent was successful in adapting its strategies based on the given problem configuration. The authors further discussed how this learning behavior was achieved, with a focus on the agent's state, activity, and incentive design.

B. Multi-Agents

The robots that work in the marine environment are Unmanned surface vehicles (USV), which is a surface ship that can detect the target, and do the perception of the environment.

TABLE I. IN THE LITERATURE REVIEW TABLE, THE FEATURE THAT HAS NOT BEEN ADDRESSED IN PREVIOUS WORK IS DENOTED BY ×, WHILE √ INDICATES THAT THE FEATURE HAS BEEN ADDRESSED IN PRIOR RESEARCH

Ref.	Multi Agents	Agents make joint actions	Single Agent	Partially observable environment	Real world complex environment
[5]	√	√	√	×	×
[11]	×	×	√	×	×
[20]	×	×	√	×	√
[12]	×	×	√	×	×
[29]	×	×	√	×	×
[30]	√	×	×	×	×
[34]	×	×	√	×	×
[35]	√	×	×	×	×
Proposed Model	√	√	√	√	√

Autonomous navigation and obstacle avoidance are very necessary for USV and have been frequently employed to carry out operations in a complex sea environment or in risky marine areas for ships with sailors, considerably increasing their defensive capability and detection range [26] [27]. Traditional navigation approaches increase the risk of falling into traps while travelling in a complex, dynamic environment, and the likelihood of reaching the target decreases. These heuristic approaches are slow, and when used in real-world sophisticated dynamic situations, they have issues such as slow speed and inability to identify and avoid obstacles [24] [25]. Authors proposed a DRL approach called ANOA (Autonomous Navigation and Obstacle Avoidance), which uses a dueling deep Q-network and a customized architecture of state and action spaces [20]. They implemented their proposed approach in a static and dynamic environment using Unity 3D and the ML-Agents Toolkit. Their experimental results show that once they are trained, USVs can find a path, reach their destination, and avoid obstacles in any complex dynamic environment. In [21], authors proposed a DRL based framework for the collaboration and cooperation of multi agents in real-world environments. A partitioned base technique is presented in this work. For effective coordination and continuous operation of a multi-agent system, it is necessary for agents to have knowledge of the environment's boundary and the location of the currently active neighboring agent. This allows for seamless transition to the next neighboring agent in case of any unexpected issue or failure.

Authors explore the feasibility of utilizing the PPO, a cutting-edge DRL algorithm for continuous control tasks, to address the challenge of effective path planning that caters to the two primary objectives of path following and collision avoidance (COLAV), essential for ensuring the safety and dependability of autonomous surface vehicles (ASV) [29]. The AI agent is trained and evaluated in a complex, stochastically generated simulation environment employing the OpenAI gym Python toolbox equipped with several rangefinder sensors for obstacle detection. The agent has real-time visibility into its own reward function, which enables it to adapt its guidance strategy dynamically. The trained agent achieves an episodic success rate of almost 100 percent, varying from path-adherence to obstacle avoidance.

The application of deep neural network-based systems has become a state-of-the-art approach in various robotics tasks. However, utilizing these systems in safety-critical areas can be precarious in the absence of explicit network robustness guarantees [22] [23]. Small changes in sensor inputs (due to noise or adversarial situations) can cause network-based decisions to change, as demonstrated by an autonomous car swerving into another lane. To mitigate the risks posed by adversarial attacks, various defensive algorithms have been developed, including those that provide formal robustness guarantees. In the study [30], certified adversarial robustness research is utilized to develop an online DRL method that is certifiably robust. The proposed defense mechanism calculates lower bounds on state-action values to detect and select a resilient action in case of worst-case input divergence caused by possible adversaries or noise. Additionally, the resulting policy is accompanied by a certificate of solution quality, despite the certifier's lack of knowledge about the real state and optimal action due to disturbances. The study provides new performance guarantees, extensions to other RL algorithms, aggregated results across multiple scenarios, an extension into scenarios with adversarial behavior, comparisons with a more computationally expensive method, and visualizations that provide intuition about the robustness algorithm [31] [32].

In Table I, the existing literature is comprehensively summarized. They must guarantee that DRL-trained robots are robust in order for them to function alongside humans in actual circumstances. Because the current world is so diverse, and human behaviour frequently changes in reaction to agent deployment, the agent will almost certainly encounter scenarios that he or she has never encountered before. This creates an assessment challenge: how can authors successfully measure robustness if they can't use the average training or validation reward as a metric? Unit testing in software engineering is a source of inspiration for them. In their paper [34], the authors propose that when creating AI agents that interact with people, designers should consider potential edge cases in partner behavior, conceivable states encountered, and construct tests to guarantee that the agent's behaviour is appropriate in these instances. This technique is used to create a suite of unit tests for the Overcooked-AI environment, which is then used to assess three ideas for enhancing robustness. Authors discover that using the test suite gives them a lot of information about the consequences of these proposals that they couldn't get by just looking at the average validation reward.

III. JUSTIFICATION OF THE PROPOSED MODEL

It would be naive to assert that MAS should be employed in all complex system designs. There are certain cases when it is especially suited, as with any beneficial strategy, and others where it is not.Multi-Agent Systems (MAS) have numerous real-world applications, including but not limited to traffic control, task distribution, ant colonies, autonomous cars, and drones. A hard-coded agent cannot operate well in a high-dimensional environment due to the complexity of the environment. MADRL agents can find the best solutions and policies. The policies of other agents are impacted by changes in the environment when agents collaborate and coordinate their actions. Failure of a learnt policy in safety-critical applications might result in dangerous scenarios. The purpose of this section is to emphasise the importance and utility of MADRL in accordance with safety. In safety-critical

applications, ensuring the robustness of policies is essential when using RL to solve problems. To enhance the safety of learned policies, control approaches from the control community can be utilized. Safety-critical scenarios are not considered in multi-agent settings. So, there is a need to develop a DRL-based system that tackles the problem of safety in a multi-agent environment.

IV. Proposed Model

A technique for approximating the safety robustness of policies is proposed, which is based on the system architecture of robust control, safe states (i.e., non-error states [35]), and statistical analysis. The safety robustness is approximated by calculating the ratio of the number of safe episodes to the total number of episodes, which can be used as an indicator to evaluate policy robustness from a safety perspective. This model uses curriculum learning for the randomization of the environment and the better performance of agents. Agents start their learning with easy tasks and then move to the difficult ones. We use PPO to analyze the visual observations as well as vector observations. By using visual and vector observations, agents communicate with each other. In order to train our model, we employ a 4-dimensional action space in the environment. CNN is the neural network used by PPO (a fully linked neural network).

The environment is sensed by the agent's sensors, which are subsequently passed on to the first layer of the neural network. The activation function is used here before moving on to the next layer. Similarly, the agent receives the output after using the activation function. The agent receives this output and estimates his action based on that output. The agent receives a positive reward if it takes the correct action, while a negative reward is given if it takes the wrong action.

We also use CL (a method of training a machine learning model that gradually introduces more challenging components of a problem such that the model is constantly optimally challenged) for training purposes also.

A. Proposed Method of DRL

When agents interact with the environment, there are a number of observations made by them. According to observations, an agent must follow some policy and, as a result, must take action [28]. By taking that action, the agent makes different changes in the environment. Changes may be positive (perform the best action to tackle the task) or may be negative. (perform bad actions, exploit the task or environment). After performing an action, the agent moves to another state. The agent again observes the environment and follows another policy to gain the optimal policy for that task.

We employ the PPO algorithm for training in our multi-agent settings, and by combining it with CL, we can expand the training process to a large number of agents. The learning environment is entirely cooperative, and each agent possesses only basic knowledge of the surroundings. The goal of all agents is to maximize the total discounted reward by working together. The following is a description of a tuple:

$$(N, S, A_i, Z_i, C, R, O)$$

N denotes the total number of agents in the system, S represents the number of possible states that the system can be in, and A_i denotes the number of possible actions that agent i can take. C and R represent the joint transition and reward functions, respectively. O represents the number of possible observations that can be made, and Z_i represents the collection of observations that agent i has access to.

To make the training more stable, each agent only knows limited knowledge about the environment in a single state S_c. As a result, agents must recall some information from prior observations.

The utilization of the PPO algorithm was employed to improve training with limited rewards. This method involves the use of a conditional statement to differentiate between actions and observations derived from the agent's demonstration. The condition assesses whether the new actions and observations are similar to those in the demonstration. The agents are rewarded for their efforts. We may also take advantage of local incentives using PPO. This investigation employs an extrinsic reward signal as a local reward signal, and utilizes a combination of CL with RL and PPO to expedite policy convergence. Fig. 1 is the demonstration of the proposed system model.

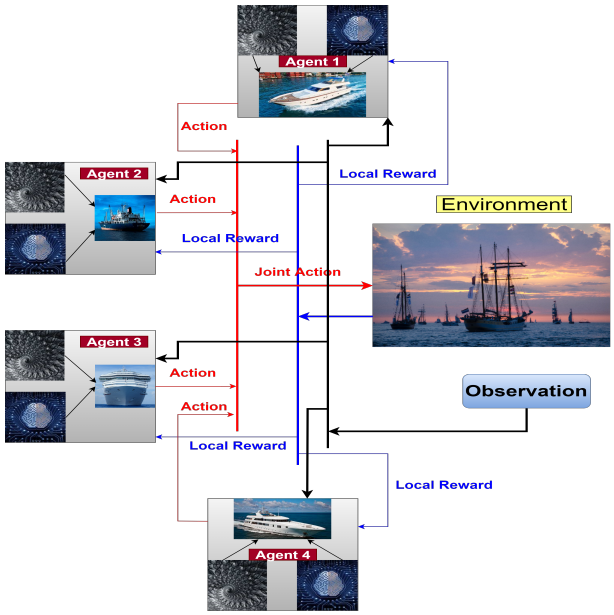

Fig. 1. Block diagram of the proposed system model [33].

In Fig. 2, obtained from [33], you can see that agents observe visual and vector observations from the environment and move these inputs (observations) to the convolutional layer. The agent receives the outputs of the results. According to these outputs, agents take action in the environment.

The mentioned techniques are implemented for each agent, which operate by collaborating and cooperating with other agents. The agents interact with the learning environment through cooperative actions, and as a result, they receive a local cumulative reward.

This allows agents to quickly imitate demonstrated behaviour, which speeds up the training process. The CL tech-

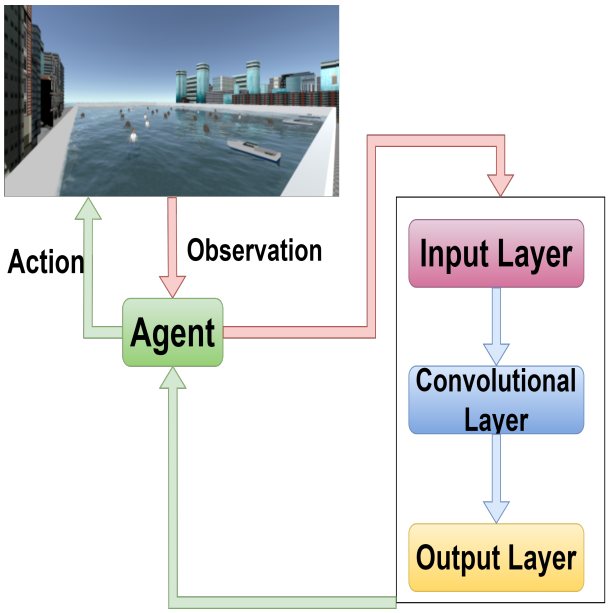

Fig. 2. Layered approach in the proposed system model.

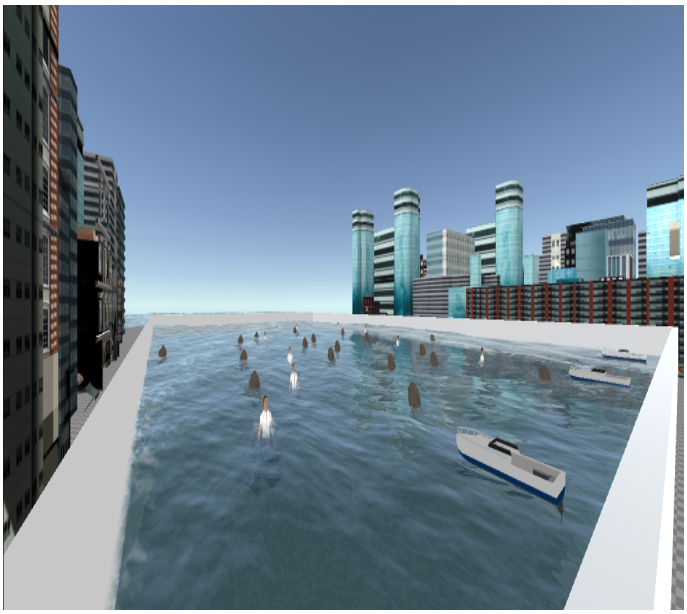

Fig. 3. A scene of the learning environment for agents.

nique guarantees that the policies are converged quickly. The algorithm provides the pseudo-code for the suggested model.

B. Components of the Proposed Model

Different components of proposed model are as under:

1) Learning environment: In many aspects of life, robots play an important role. Recently, robots have been used in digital factories [36], as well as in hazardous locations such as fires, storms, and landslides. Robots can carry, move, and lift large things from one location to another, making it easier to provide goods to people who are in risky or challenging situations. Robots operate more effectively and quickly than people in difficult situations, reducing human risk. These well-trained agents have also been utilised in the marine environment to rescue individuals and apprehend smugglers. As a result, we decided to learn in a maritime setting. This paper looks at a real-world situation in which agents try to save people by getting around problems and working together.

The learning environment employed in this study is depicted in Fig. 3. Agents in a marine setting navigate the goal (people in need of rescue) and navigate to them while avoiding obstacles. For example, the environment contains obstacles in the form of heavy rocks, and the agents have the ability to move in four directions: left, right, forward, and backward. Agents go towards their target by doing these activities and avoiding obstacles through navigation and coordination with other agents.

2) Agents: In Unity 3D games, objects are used as agents in the learning environment. The agent is a boat that is used to move in a marine environment and to rescue individuals who are drowning. In Fig. 4 agent is demonstrated as "ship".

In the learning environment, agents generate states and execute actions from a set of predefined actions such as left, right, forward, and backward, thereby transitioning from one state, denoted by S_t to the next state, denoted by S_{t+1}. Each agent is associated with a single brain that guides the agent on its next course of action.

Fig. 4. A scene of an agent navigating in the given environment.

C. Training Process Algorithm

In Algorithm 1, it is observed that the process starts from episodes. Here we apply a "For-loop" that initiates the training process from the first episode.

Algorithm 1 Training Process

For episodes = 1 to n
{
Initialize the Environment
Randomize the Environment
For Time-steps = 1 to 5000
{
For each Agent $a_i < 4$
Agent a_i observe Environment
Calculate distance of every agent
{
if (Distance <= 50 meters of Agent a_i AND Agent a_j)
{
Broadcast message to Agent a_j
}
if (Goal of Agent a_i AND Agent a_j == same)
{
Agent with minimum distance from goal will take action.
}
}
Calculate Reward.
}
Mean Reward = *Actions with positive reward / Total Reward*
if (Mean Reward > 0)
{
Safe Episode
}
}
Safety Robustness of Current Policy = *total Safe Episodes / total Episodes*

Episode and then moves on to the next to meet the desired number of episodes. The environment is initialised in each episode. After initialization, the environment is randomized. Because the examples of dynamic environments are considered so, there is a need to observe all the environments. After randomizing the environment, another "For-loop" is applied inside the first for-loop. This for-loop starts from the first time-step to the 5000 time-steps of each episode. Within this loop, there is an additional nested "for-loop" that iterates through each agent in the environment.

In each time-step, every agent observes the environment and calculates their distance from other agents. Here is the if statement. If the goals of both agents are common, then the agent having the minimum distance from the goal will take action and collect a reward while the other agent will select the other target/goal. If both agents have different goals, both agents will perform or take their actions according to their targets. When 1 episode is completed, the mean reward will be calculated by dividing the total positive reward by the total reward. The following "if condition" checks if the mean reward is above zero. If this condition is satisfied, the episode is considered safe. After all episodes are completed, the policy's safety and robustness are determined by calculating the ratio of safe episodes to total episodes.

D. Flow Chart

The flow chart of our proposed model is shown in Fig. 5. It is observed that in each episode, the environment is initialized. Because after initialization, the environment is randomized. because the examples of dynamic environments are considered. Here is a loop. If time steps are completed, the process will start from another episode. If the time-steps are pending, then in each time-step, every agent observes the environment. According to that observation, agents calculate their distance from other agents. After calculating the distance, agents having a minimum distance of 50 metres broadcast their goals. If the goals of both agents are common then the agent having minimum distance from the goal will take action and collect reward and the other agent will select the other target/goal. If both agents have different goals, both agents will perform their actions to meet their target. After completing one episode, the mean reward can be calculated by dividing the total positive rewards obtained during the episode by the total number of rewards received. Completion of all episodes results in the safety and robustness of the current policy.

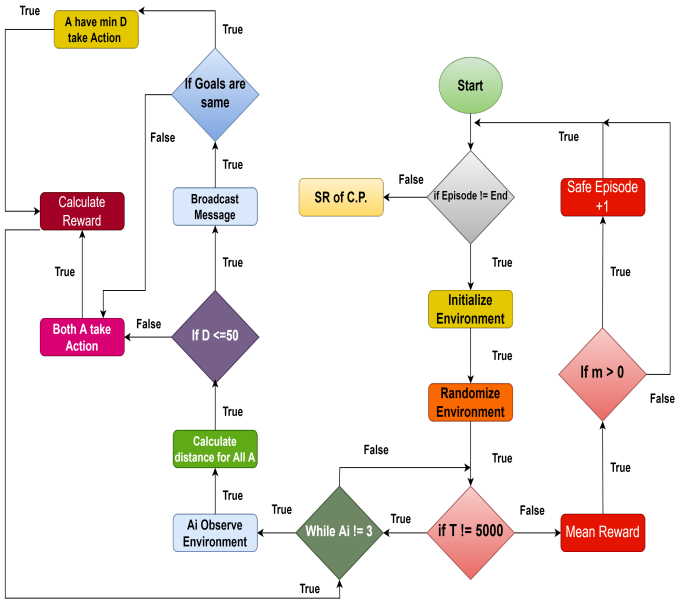

Fig. 5. The flow chart of the proposed system.

E. Reward Function

To promote desirable behavior, a local incentive signal is utilized to encourage consistency. An effective incentive system ensures faster convergence of the learning process in single-agent contexts than ever before [37]. The entire reward for a single agent may be calculated using the equation below:

$$r = (r_{hp} + r_{hc}) - (r_{cla} + r_{clo} + r_{cldo})$$

where, r is the sum of rewards given to each agent, where r_{hp} is the reward for rescuing drowning humans, r_{hc} is the reward for rescuing all humans in the environment, r_{clo} is the negative reward for colliding with an obstacle, r_{cla} is the negative reward for colliding with other agents and (r cldo) is the negative reward for colliding with some dynamic or sudden obstacle. r_{hp} can be calculated using the following equation:

$$r_{hp} = \begin{cases} r_p & \text{if} \quad d = 0 \\ 0 & \text{Otherwise} \end{cases}$$

where, r_p is the reward for rescuing people and 'd' represents the distance between the agent and the drowning person.

Moreover, we can calculate reward r_c using the following equation:

$$r_{hc} = \begin{cases} r_c & \text{if} \quad h = 0 \\ 0 & \text{Otherwise} \end{cases}$$

where, h_c is the reward of rescuing all drowning people, and h is the number of drowned people remaining in the sea.

Furthermore, to calculate the r_{cla} reward, we can use the following equation:

$$r_{cla} = \begin{cases} r_a & \text{if} \quad d = 0 \\ 0 & \text{Otherwise} \end{cases}$$

the reward when an agent collides with another agent is denoted by r_a and d is the distance between the two agents.

Moreover, to calculate the r_{clo} reward, we present the following equation:

$$r_{clo} = \begin{cases} r_o & \text{if} \quad d = 0 \\ 0 & \text{Otherwise} \end{cases}$$

where, r_o is the reward given to the agent on collision with static obstacles, and d is the distance between the agent and the static obstacle.

Lastly, we present the equation to calculate the r_{cldo} reward.

$$r_{cldo} = \begin{cases} r_{do} & \text{if} \quad d = 0 \\ 0 & \text{Otherwise} \end{cases}$$

where r_{do} is the reward of an agent's collision with a moving drowning human or any dynamic obstacle, and d is the distance between the agent and the drowning human or dynamic obstacle.

The table below illustrates the awards each agent received for various occurrences. We implemented a penalty of -0.1 for each agent when they collided with another agent in the environment. This approach addresses the issue of inter-agent collision. We also assign a value of -0.1 when the agent collides with the seashore or any other obstruction. The agent earns-0.1 reward if it collides with any drowning human in the surroundings. An agent who successfully rescues a drowning individual receives a +0.3 bonus. Furthermore, the agents will gain a +0.5 bonus if they are successful in rescuing all drowning people in the surroundings.

V. Implementation

This section describes how we will put the suggested approach into practice. We also go through the tools that were utilised to carry out the implementation. This paper also investigates various behavioral characteristics and training parameters.

A. Unity 3D

Unity Technologies is the developer of the cross-platform game engine called Unity, which was first announced and released as a game engine for Mac OS X in June 2005 at Apple Inc.'s Worldwide Developers Conference. Over time, the engine has been enhanced to support various platforms such as PC, mobile, console, and virtual reality. It allows developers to create 3D and 2D games, interactive simulations [38], and other experiences.

Gazebo is also a three-dimensional dynamic robot environment simulator that can be used to test algorithms and robot designs. It runs regression tests and uses realistic scenarios to teach AI systems. The speed of the Gazebo Simulator has never been a strong point. Simulating a 30-minute event takes 6 hours. Multiple instances operating at the same time is tough. We can make it happen, but there are still difficulties like non-accuracy and an unknown amount of effort required to coordinate among these various instances.

Unity [39], on the other hand, is far more efficient at running several simulations on a single system and coordinating learning across all instances. Unity simulations may be cranked up to a hundred times faster than real time.

B. ML-Agents Toolkit

The toolbox includes numerous components for implementing RL, such as Brain, Academy, and Agent. Each agent is connected to and controlled by a brain component. Different environmental factors are learned using the Academy component. Additionally, it provides a link between the brain and Python Tensorflow. Fig. 6 shows the basic structure and overview of the ML-Agents toolkit [40].

C. Training Parameters

The learning environment's working area is 200×200 square feet. The functional area of the learning environment contains a variety of objects, including rocks, boats, humans, and agents. The combination of moving individuals and robots was utilized to enhance the realism and dynamism of the learning environment, enabling the agents to be trained in such a setting.

TABLE II. Training Parameters Used in the Proposed Model

Parameters	Values
Batch size	1024
Buffer size	16384
Learning rate	0.0003
Max steps for each episode	5000
Number of epoch	3
Number 0f layers	2
Gamma in extrinsic Reward signal	0.99
Behavior cloning strength	0.5

Fig. 6. ML-Agents toolkit used in the proposed model.

Table II displays some important training parameters employed to train the agents utilizing the proposed model. Based on empirical testing, a batch size of 1024 is deemed appropriate. Additionally, the buffer size parameter must be larger than the batch size. The buffer size was set to 16384 by default. Local RL reward signals are used in the proposed model.

TABLE III. TRAINING PARAMETERS USED FOR POLICY 1

Parameters	Values
Batch size	1024
Buffer size	16384
Learning rate	0.0003
Max steps for each episode	5000
Number of epoch	3
Number 0f layers	2
Gamma in extrinsic Reward signal	0.99
Time horizon	128
Hidden Units	512

The Table III provides some key training parameters for policy 1, which were utilized to train the agents. The batch size for the PPO algorithm has been adjusted to 1024. By default, we set the buffer size parameter to 16384, which is greater than the batch size of 1024. For policy 1, we set the temporal horizon value to 128 and the hidden units value to 512.

The Table IV provides some key training parameters for policy 2, which were utilised to train the agents. The batch size for the SAC algorithm has been adjusted to 1024. By default, we set the buffer size parameter to 16384, which is greater than the batch size of 1024. For the SAC algorithm, we set the temporal horizon to 128 and the hidden unit value to 512. Initial buffer steps of 6000 are used. Tua is used at a value of 0.01. For policy 2, we employ 10.0 steps for each update.

TABLE IV. TRAINING PARAMETERS USED FOR POLICY 2

Parameters	Values
Batch size	1024
Buffer size	16384
Learning rate	0.0003
Max steps for each episode	5000
Number of epoch	3
Number 0f layers	2
Gamma in extrinsic Reward signal	0.99
Time horizon	128
Hidden Units	512
Tau	0.01
Buffer initial steps	6000
Steps per update	10.0

VI. EXPERIMENTS AND RESULTS

With the aforesaid configuration, Algorithm 1 is used to evaluate the safety robustness of the three approaches for managing USVs in terms of obstacle avoidance and navigation to their destination. Fig. 7 illustrates the orientation of the USVs controlled by the three policies (p1, p2, and p3) towards a goal orientation when the maximum uncertainty of action and state is 0.1. It can be observed that all three policies successfully maintain the USVs in safe states.

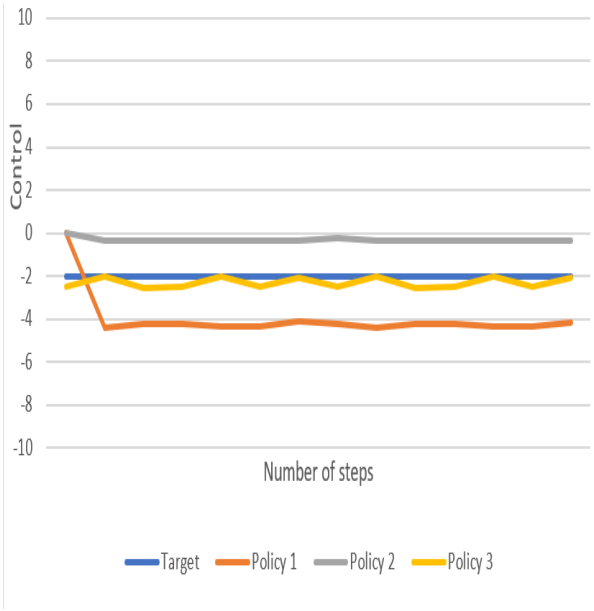

Fig. 7. Orientation of the USVs controlled by the three policies with action uncertainty value 0.1.

The y-axis depicts the control of USVs, while the x-axis represents the number of time-steps.

The target is shown by a blue line, the policy-1 is indicated by a red line, the policy-2 is indicated by a grey line, and the policy-3 is indicated by a yellow line.

The proposed model's policy is policy-3, and it can be observed that policy-3 is closer to the aim than the other policies. It can be seen from Fig. 7, the policy followed by our proposed model is more resilient in terms of USV's control than the other policies.

When the maximum magnitudes of the uncertainties of action and state are 0.2, Fig. 8 shows the results.

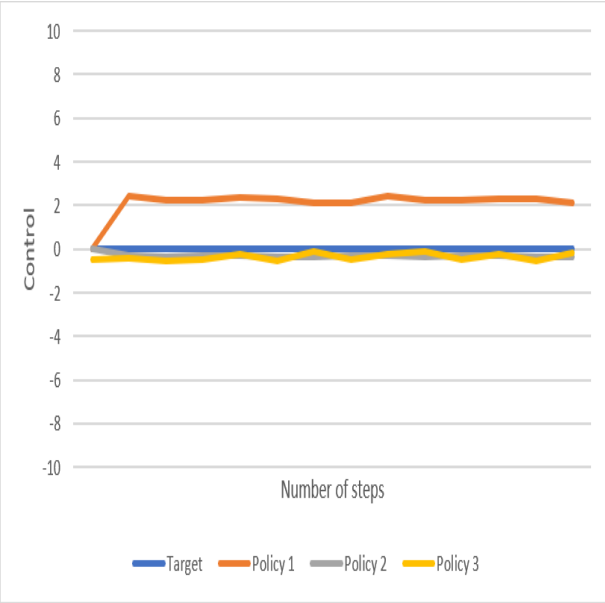

Fig. 8. USV control with action uncertainty of 0.2.

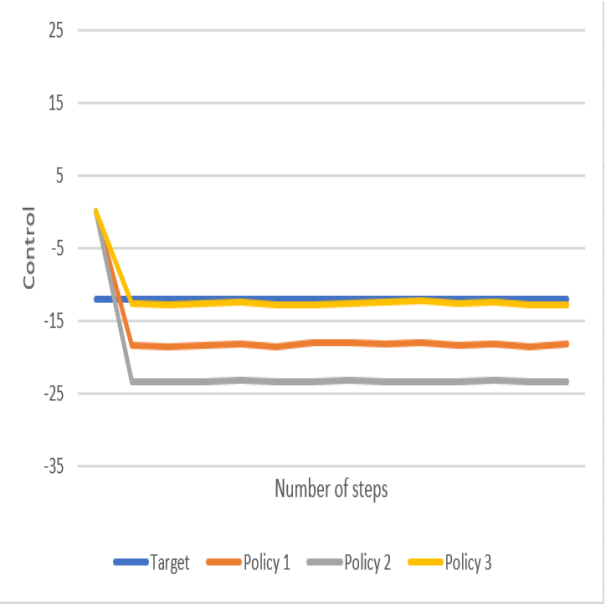

Fig. 9. USV control with action uncertainty of 0.5.

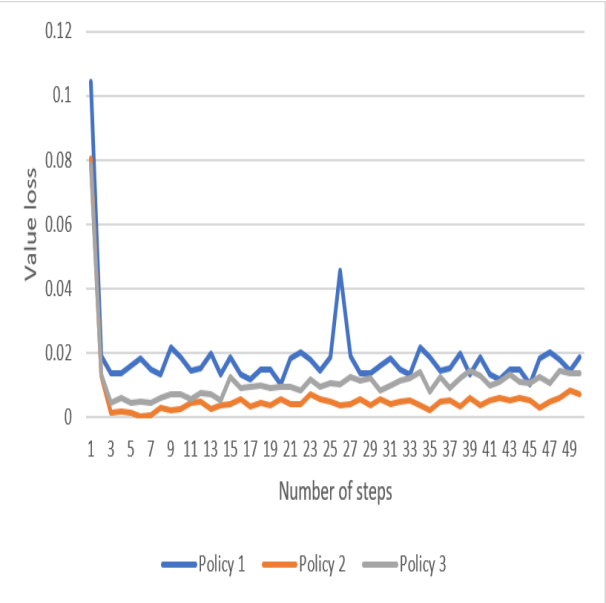

Fig. 10. Value loss against the number of steps in 3 different policies.

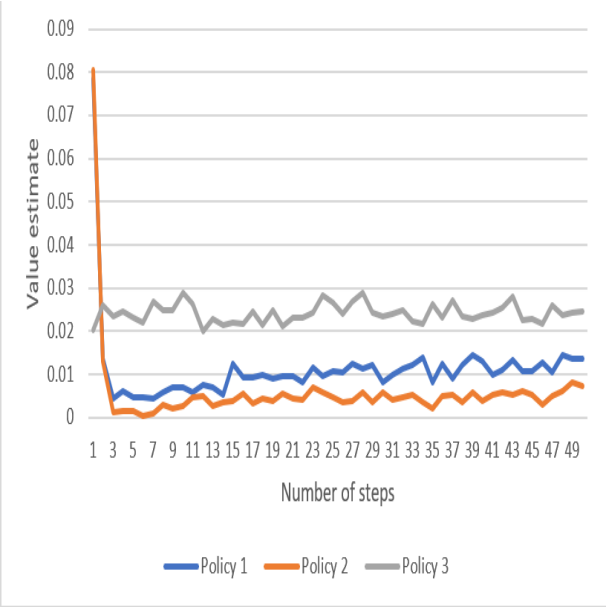

Fig. 11. Value estimate against the number of steps in 3 different policies.

The USV control is represented on the y-axis, while the number of time-steps is represented on the x-axis.

The plot displays the target as a blue line, while the policy-1, policy-2, and policy-3 are represented by a red, grey, and yellow line, respectively.

The proposed model's policy is policy-3, and it can be observed that policy-3 is closer to the aim than the other policies.

It can be seen from Fig. 8, the policy followed by our proposed model is more resilient in terms of USV's control than the other policies.

Fig. 9 illustrates the orientation of USVs controlled by the three policies (p1, p2, and p3) to a goal orientation, when the maximum magnitudes of the uncertainty of action and state are 0.5. All three policies were able to maintain the USVs in safe states.

The y-axis depicts the control of USVs, while the x-axis represents the number of time-steps.

The blue line represents the target, while the red, grey, and yellow lines represent policy-1, policy-2, and policy-3, respectively.

The episode is stopped at the 40th step when the USV is controlled by p1 since it is out of safe condition. The USV

can be kept safe by policies p2 and p3.

As a result, the three policies have varying degrees of ability to keep the USV in a safe state. Before the policy with the best safety robustness, it is crucial to evaluate the safety robustness of the three policies using Algorithm 1.

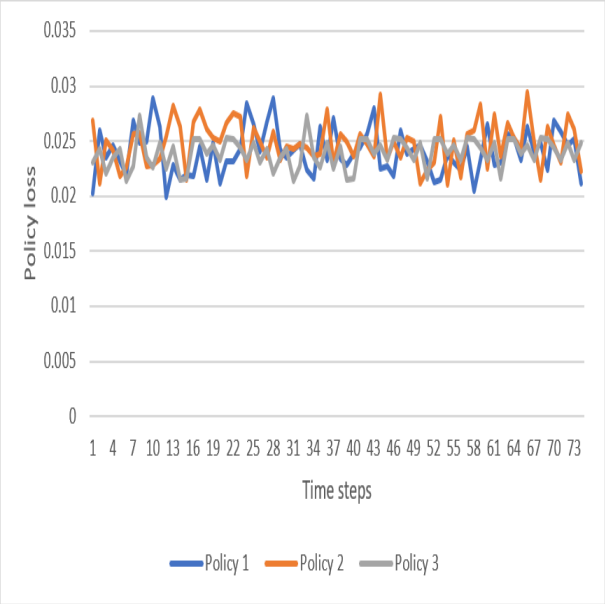

Fig. 12. Policy loss against the number of steps in 3 different policies.

In Fig. 10, the y-axis represents the value loss, while the x-axis indicates the number of time-steps. The blue line represents policy-1, the orange line represents policy-2, and the grey line represents policy-3. The graph indicates that policy-3 has an intermediate value loss between policy-1 and policy-2, suggesting that policy-3 has an average value loss.

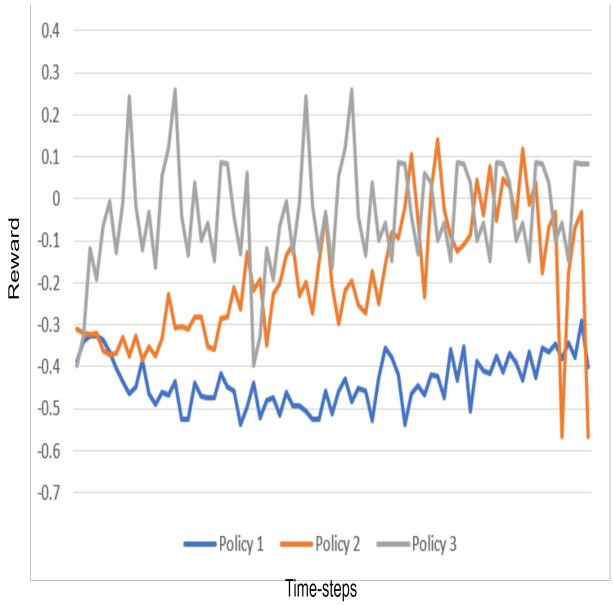

Fig. 13. Reward against the number of steps in 3 different policies.

Fig. 12 plots the value loss against the number of time-steps, where the y-axis represents the value loss, and the x-axis represents the time-steps. The performance of the three policies (i.e., policy-1, policy-2, and policy-3) is shown in blue, orange, and grey lines, respectively. The figure demonstrates that policy-3 has the lowest value loss among the three policies.

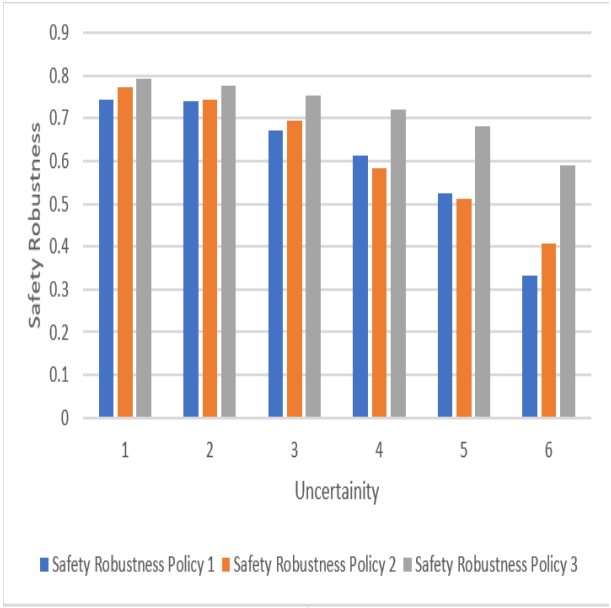

Fig. 14. Safety robustness vs. Uncertainty in 3 different policies.

The plot in Fig. 11 shows the value loss on the y-axis against the number of time-steps on the x-axis. The three policies, Policy 1, Policy 2, and Policy 3, are represented by blue, orange, and grey lines, respectively. It is evident from the plot that Policy 3 has the highest policy loss among the three.

The graph in Fig. 13 illustrates the rewards obtained by all policies, where Policy 1 is represented by a blue line, Policy 2 by an orange line, and Policy 3 by a grey line. The reward is a measure of policy efficiency, and the policy with the highest reward indicates better efficiency. From the graph, it is evident that Policy 2 gains more rewards compared to Policy 1, and Policy 3 gains more rewards than both Policy 1 and Policy 2. The proposed model adopts Policy 3, which is demonstrated to be the most efficient among the three policies.

In Fig. 14, the y-axis represents the reward gained by all policies, while the x-axis indicates the number of episodes. The blue line represents Policy-1, the orange line represents Policy-2, and the grey line represents Policy-3. The safety robustness of a policy indicates its ability to keep the system in a safe state.

Efficiency of a model is indicated by its safety robustness, and the policy that has maximum safety robustness is considered the most efficient. In Fig. 14, it can be observed that the proposed model follows policy 3, which has the highest safety robustness among all policies.

VII. Conclusions

In this work, we investigated a DRL based scenario which use multi-agents. These agents have been assigned some tasks

related to real world scenario. In our case, a submarine scenario was considered and the task of the agents is to safe the drowning person. We extended our previously published work by adding more experiments such as policy loss, reward function and the safety robustness. While the training of agents using deep reinforcement learning in real-world scenarios is very important, the safety of agents and other human beings is also very important. It is very important to learn about a policy that is robust in the context of safety. Several previous works in single agent settings focused on the safety of agents and other aspects of their training environment. But no work was done on the robust safety of agents in multi-agent settings. This work is focused on the safety and robustness of policies in multi-agent settings. A model is proposed that ensures the safety of agents and other valuable entities in the environment. The proposed model uses PPO algorithm to train the agents. Curriculum learning is used to teach the agent about safety and find a robust policy from easy to difficult level. A sea environment is considered in this work where some USVs are rescuing human beings that have been dropped into the sea. An algorithm is proposed in this research for the training of agents and finds a robust policy in the context of safety. To evaluate the safety robustness of policies, a few parameters are evaluated in this work. Safety robustness with respect to the number of safe states and control of USVs for the target with respect to the number of steps is evaluated, which shows the robustness of policy in safety. The policy followed by the proposed model is compared with the other two policies. Through experimental evaluation, it has been observed that the proposed model outperforms other policies. Furthermore, the performance of the proposed model has been verified by evaluating parameters such as mean reward, policy loss, policy estimate, and value loss. The experimental results have demonstrated that the proposed model exhibits better performance in multi-agent scenarios.

In the future, we aim to test our model with a higher number of agents in more complex scenarios. For example, it is vital to investigate how wind and its speed can effect the safety and robustness of the agents in the DRL scenario.

ACKNOWLEDGMENTS

The authors would like to acknowledge the support from Department of Neteorks and Communications, College of Computer Science and Information Technology, and Deanship of Scientific Research (DSR), King Faisal University, Al-Ahsa, Kindom of Saudi Arabia.

REFERENCES

[1] E. L. Thorndike, Animal intelligence: An experimental study of the associate processes in animals., American Psychologist 53 (10) (1998) 1125.

[2] etelli, M. Pirotta, M. Restelli, Compatible reward inverse reinforcement learning, in: The Thirty-first Annual Conference on Neural Information Processing Systems-NIPS 2017, 2017.

[3] K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Processing Magazine 570 34 (6) (2017) 26–38.(2020) 230–240.

[4] N. D. Nguyen, T. Nguyen, S. Nahavandi, System design perspective for human-level agents using deep reinforcement learning: A survey, IEEE Access 5 (2017) 27091–27102.

[5] E. A. O. Diallo, A. Sugiyama, T. Sugawara, Coordinated behavior of cooperative agents using deep reinforcement learning, Neurocomputing 396.

[6] Y. Dong, X. Tang, Y. Yuan, Principled reward shaping for reinforcement learning via Lyapunov stability theory, Neurocomputing 393 (2020) 83–90.

[7] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, Domain randomization for transferring deep neural networks from simulation to the real world, in: 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), IEEE, 2017, pp. 23–30.

[8] M. Yan, I. Frosio, S. Tyree, J. Kautz, Sim-to-real transfer of accurate grasping with eye-in-hand observations and continuous control, arXiv preprint 585 arXiv:1712.03303.

[9] Z. Tang, S. Liu, G. Bommannan, G. Chowdhary, Robust deep reinforcement learning with adversarial attacks, arXiv preprint arXiv:1712.03632.

[10] L. Pinto, J. Davidson, R. Sukthankar, A. Gupta, Robust adversarial reinforcement learning, in: International Conference on Machine Learning, PMLR, 2017, pp. 2817–2826.

[11] H. Xiong, X. Diao, Safety robustness of reinforcement learning policies: A view from robust control, Neurocomputing 422 (2021) 12–21.

[12] V. J. Hodge, R. Hawkins, R. Alexander, Deep reinforcement learning for drone navigation using sensor data, Neural Computing and Applications 33 (6) (2021) 2015–2033.

[13] L. Berducci, S. Yang, R. Mangharam and R. Grosu, "Learning Adaptive Safety for Multi-Agent Systems," 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 2024, pp. 2859-2865, doi: 10.1109/ICRA57147.2024.10611037.

[14] Z. Lv, L. Xiao, Y. Chen, H. Chen and X. Ji, "Safe Multi-Agent Reinforcement Learning for Wireless Applications Against Adversarial Communications," in IEEE Transactions on Information Forensics and Security, vol. 19, pp. 6824-6839, 2024, doi: 10.1109/TIFS.2024.3423428.

[15] T. T. Nguyen, N. D. Nguyen, S. Nahavandi, Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications, IEEE transactions on cybernetics 50 (9) (2020) 3826–3839.

[16] Q. Yao, Z. Zheng, L. Qi, H. Yuan, X. Guo, M. Zhao, Z. Liu, T. Yang, Path planning method with improved artificial potential field—a reinforcement learning perspective, IEEE Access 8 (2020) 135513–135523.

[17] D.-H. Chun, M.-I. Roh, H.-W. Lee, J. Ha, D. Yu, Deep reinforcement learning-based collision avoidance for an autonomous ship, Ocean Engineering 234 (2021) 109216.

[18] H. Zhang, H. Chen, D. Boning, C.-J. Hsieh, Robust reinforcement learning on state observations with learned optimal adversary, arXiv preprint arXiv:2101.08452.

[19] S. Mayer, T. Classen, C. Endisch, Modular production control using deep reinforcement learning: proximal policy optimization, Journal of Intelligent Manufacturing (2021) 1–17.

[20] X. Wu, H. Chen, C. Chen, M. Zhong, S. Xie, Y. Guo, H. Fujita, The autonomous navigation and obstacle avoidance for usvs with anoa deep reinforcement learning method, Knowledge-Based Systems 196 (2020) 105201.

[21] V. J. Hodge, R. Hawkins, R. Alexander, Deep reinforcement learning for drone navigation using sensor data, Neural Computing and Applications (2020) 1–19.

[22] Y. Wang, H. He, C. Sun, Learning to navigate through complex dynamic environment with modular deep reinforcement learning, IEEE Transactions on Games 10 (4) (2018) 400–412.

[23] H. Shi, Z. Lin, K.-S. Hwang, S. Yang, J. Chen, An adaptive strategy selection method with reinforcement learning for robotic soccer games, IEEE Access 6 (2018) 8376–8386.

[24] J. Meng, F. Zhu, Y. Ge, and P. Zhao, "Integrating safety constraints into adversarial training for robust deep reinforcement learning," Information Sciences, vol. 619, pp. 310–323, 2023, doi: https://doi.org/10.1016/j.ins.2022.11.051.

[25] S. Gu et al., "Safe multi-agent reinforcement learning for multi-robot control," Artificial Intelligence, vol. 319, p. 103905, 2023, doi: https://doi.org/10.1016/j.artint.2023.103905.

[26] S. Reis, L. P. Reis, N. Lau, Game adaptation by using reinforcement learning over meta games, Group Decision and Negotiation (2020) 1–20.

[27] M. Luong, C. Pham, Incremental learning for autonomous navigation of mobile robots based on deep reinforcement learning, Journal of Intelligent Robotic Systems 101 (1) (2021) 1–11.

[28] A. Raza, M. A. Shah, H. A. Khattak, C. Maple, F. Al-Turjman, H. T. Rauf, 630 Collaborative multi-agents in dynamic industrial internet of things using deep reinforcement learning, Environment, Development and Sustainability 24 (7) (2022) 9481–9499.

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

[30] M. Everett, B. L¨utjens, J. P. How, Certifiable robustness to adversarial state uncertainty in deep reinforcement learning, IEEE Transactions on Neural Networks and Learning Systems.

[31] J. Meng, F. Zhu, Y. Ge, and P. Zhao, "Integrating safety constraints into adversarial training for robust deep reinforcement learning," Information Sciences, vol. 619, pp. 310–323, 2023, doi: https://doi.org/10.1016/j.ins.2022.11.051.

[32] S. Gu et al., "Safe multi-agent reinforcement learning for multi-robot control," Artificial Intelligence, vol. 319, p. 103905, 2023, doi: https://doi.org/10.1016/j.artint.2023.103905.

[33] M. A. Shah, "Safety robustness of deep reinforcement learning in multi-agent scenarios," IET Conference Proceedings, vol. 2025, no. 22, pp. 34–39, Oct. 2025, doi: https://doi.org/10.1049/icp.2025.2952.

[34] P. Knott, M. Carroll, S. Devlin, K. Ciosek, K. Hofmann, A. D. Dragan, 640 R. Shah, Evaluating the robustness of collaborative agents, arXiv preprint arXiv:2101.05507.

[35] T. Bansal, S. S. Agrawal, D. Kumar, M. Shambu, P. Inbarajan, Ai based diagnostic service for iot enabled smart refrigerators, in: 2021 8th International Conference on Future Internet of Things and Cloud (FiCloud),IEEE, 2021, pp. 163–168.

[36] N. Bicocchi, G. Cabri, L. Leonardi, G. Salierno, Intelligent agents supporting digital factories., in: WOA, 2019, pp. 29–34.

[37] A. Barraza-Urbina, The exploration-exploitation trade-off in interactive recommender systems, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, 2017, pp. 431–435.

[38] Y. Kuang, X. Bai, The research of virtual reality scene modeling based on unity 3d, in: 2018 13th International Conference on Computer Science Education (ICCSE), IEEE, 2018, pp. 1–3.

[39] K. Cheliotis, Abmu: An agent-based modelling framework for unity3d, SoftwareX 15 (2021) 100771.

[40] P. Cummings, A. Crooks, Development of a hybrid machine learning agent based model for optimization and interpretability, in: International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation, Springer, 2020, 660 pp. 151–160.