Autonomous Multi-Rover Guidance and Navigation for Planetary Exploration using Stacked Collaborative Deep Q-Networks

Rohaan Ahmed
Mission Systems Engineer, MDA Space
PhD Computer Science (Student), Ryerson University
Toronto, Canada
rohaan.ahmed@ryerson.ca

***Abstract*—Robotic systems operating in deep space will require high levels of autonomy in order to achieve time-efficient and collaborative exploration and exploitation. This is especially true for Planetary Exploration using Rovers. It would be a vital feature, therefore, to have on-board systems capable of providing real-time Guidance and Navigation in realistic mission scenarios. In this paper, we set out a *TRL 3 - Proof of Concept* for a Multi-Rover Guidance and Navigation System using a Stack of Collaborative Deep Q-Networks. We present our findings via two realistic Mission Scenarios; (1) where the rovers' DQNs, called Rover Agents, act independently within an environment which contains hazards, and (2) where the Rover Agents collaborate with each other, and are all simultaneously guided by a parent Deep Q-Network called the Cluster Agent. The rovers are equipped with only rudimentary sensors which provide information about their immediate vicinity, and do not have access to global data.**

I. Introduction and Motivation

Robotic systems operating in deep space will require high levels of autonomous collaboration while dealing with unexpected situations, a direct consequence of operating in the harshest of environments. The author's PhD thesis looks to solve Deep Space Robotics and Automation problems using novel AI techniques. In this regard, the project presented in this paper looks to solve one of the well known challenges in space exploration: Autonomous Multi-Rover Guidance and Navigation for Planetary Exploration.

II. Problem Statement

Planetary rovers are vehicles designed to explore the surface and shallow sub-surface of foreign celestial bodies, and are considered one of the primary tools of performing space exploration [1]. They have been used to explore the Moon, Mars, and Asteroids, and will play a vital role in forming permanent human habitations on the Lunar and Martian surfaces [2] (see Fig. 1).

The most successful planeraty rover mission in history was NASA's Mars Exploration Rover (MER) B. Launched in 2003, the rover was originally designed to last only 92 Earth days on the sufrace of Mars, but ended up exceeding far beyond its operational life, from 2004 until 2018 (Fig. 2). During its 14 year life, however, it only covered a distance of 45.16km on the surface of Mars [3]. For reference, the average time

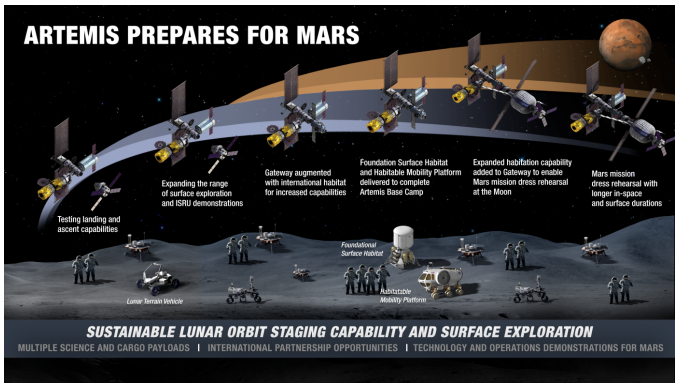

Fig. 1: NASA and ESA Artemis Mission plan for Lunar and Martian Habitation using Rovers (Credit: NASA)

it takes a trained human being to run a marathon (42km) is under 5 hours. This slow pace of exploration is a necessary constraint placed on these missions due the following factors:

- Large round-trip communication delays between Earth-based ground stations and celestial bodies, which makes real-time control of the rovers from Earth impossible. For example, round trip delay between Earth and Mars varies between 6 to 44 minutes. This .
- Limited line-of-sight communication windows between Earth and the rover, which can vary from a few hours to a few seconds per week.
- Bandwidth limitatations both due to antenna sizing and rover location.
- Signal interference due to atmospheric effects and inter-planetary electromagnetic radiation.
- Limited on-board sensors and computation, making on-board autonomy highly difficult.
- Lack of supporting infrastructure on the host celestial object, such as ground-stations, satellite imagery, terrain maps, satellite navigation, and so on.

It is obvious that if humans want to form habitations on the surfaces of foreign celestial bodies, we must increase the pace at which these rovers explore and solve problems

Fig. 2: Mars Exploration Rover - B - Opportunity (Credit: JPL-NASA)

collaboratively. To achieve this, future planetary rovers will require high levels of autonomy and collaborative supervision in order to operate efficiently in difficult, information denied environments. This paper presents a Preliminary Proof of Concept of solving this problem using Stacked Collaborative Deep Q-Networks for the Guidance and Navigation of multiple rovers with only rudimentary on-board sensing.

III. Systems Architecture

Fig 3 shows the overall System Architecture for the project and how it fits within the two Mission Scenarios.

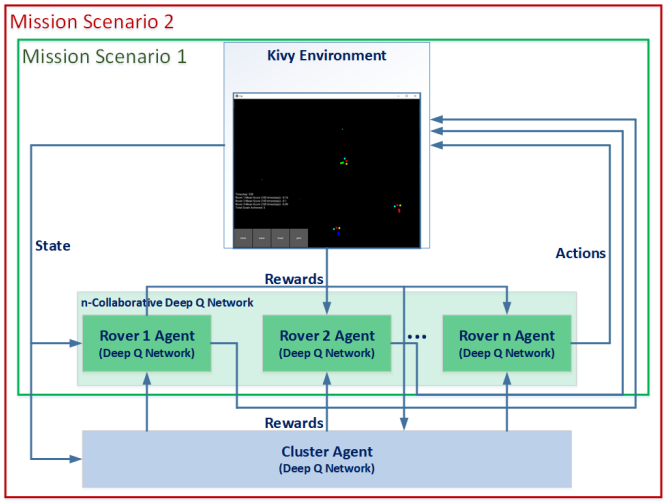

Fig. 3: System Architecture

Subsequent sections explain each of the key components in detail.

A. *Technology Readiness Level*

A central tenet of the Space Systems Engineering is incremental development of technologies per the graduated Space Technology Readiness Level (TRL) system [4]. We develop and prove our model using this approach using open source toolkits and environments, focusing on recreating realistic mission scenarios as well as possible, and leaving the fidelity of individual components for later development. This, in this project, the system been developed up to *TRL 3 - Proof of Concept*.

IV. Mission Scenarios

We developed and tested the system using different scenarios, each representing a realistic planetary exploration mission and serving as incremental building blocks. Each mission follows the following basic premise:

A. *Shared Mission Scenario Premise*

Given a number of Rovers and an equivalent number of "target" or "goal" location points within the enviroment, the agents' drive the rovers towards the goals using Deep Reinforcement Learning, while navigating their way around "sand" and each other. Once a rover reaches its goal, it is then given a new goal, and the cycle continues.

"Sand" is used as a substitute for any environmental condition where a rover must choose between proceed on its current course, or finding an alternate route. These conditions include:

- Extremely Sandy or Rocky areas where a rover may get unncecessarily bogged down.
- Rapid elevation changes, where a rover may encounter a steep incline or decline, and thus must slow down or stop.
- Any other condition which a rover may need to avoid.

To aid with their task, the rovers *only* have access to data within their immediate vicinity through on-board sensors, and do not possess any global or absolute information. Specifically, rovers must navigate within the environment using only the following information:

- Their own current position and orientation within the environment, (x_r, y_r, o_r)
- Their own current velocity, (v_x, v_y)
- Knowledge of 60 degrees in front on the vehicle
- The absolute position of their goal within the environment, (x_g, y_g)

The rover does not have access to a global map of the environment, and does not store this information on-board. As mentioned previously, this lack information is representative of true conditions in planetary exploration.

To summarize, the goal of the agents is to learn to find the most efficient path towards their objectives in an enclosed environment that includes sandy pits, while avoiding collision with other agents, using limited sensor information.

B. *Mission Scenario 1 : Multiple Rover Agents Operating Individually*

In this scenario, the rovers rely only on the local "Rover Agents" to solve the navigation problem. The rovers are unable to communicate with each other, or with a centralized control station, in order to coordinate their movements. The goal is for the rovers to reach as many goals as possible in the shortest amount of time, without colliding with each other.

This scenario represents the most realistic scenario, in the short term, for multiple autonomous rovers operating on the surface of another planet, such as Mars. Due to the lack of infrastructure, rovers would have to perform individually, while avoiding dangerous situations, such as falling into pits, entering rocky terrain, or colliding with other rovers.

C. *Mission Scenario 2: Multiple Rover Agents with Centralized Cluster Guidance*

In this scenario, the Rover Agents are able to communicate with a centralized control system housing a "Cluster Agent". The Cluster Agent controls the behaviour of all the Rover Agents by controlling the rewards each Rover Agent receives from the environment.

The goal is for the rovers to reach as many goals as possible in the shortest amount of time, without colliding with each other.

Mission Scenario 2 represents a realistic future scenario for multiple autonomous rovers operating on the surface of another planet. Having a centralized controller, either on a stationary base, such as a lander, or on-board one of the rovers, is a relatively cost effective way of providing guidance to multiple rovers at a time.

V. System Description

A. *The Environment - Kivy*

The Environment used to develop the system is Kivy. Kivy is an open source library for developing multi-touch applications across many platforms (Linux, OSX, Windows, Android, iOS), released under an MIT License. For sake of a simple analogy, it can be thought of as a *Microsoft Paint* window which can be rendered dynamically via script. Although not optimal for robotic research or object oriented design, it provides a simple and well-supported rendering environment for quick prototyping.

At each timestep, the following rendering updates are performed to the environment:

- Update the state (location and orientation) of the rovers and its sensors.
- Update the goal locations.
- Redraw the environment to reflect state updates.
- Distribute rewards to each Rover Agent based on the state update
- Communicate the new state with the appropriate Agents

It should be noted that since Kivy is not a dynamic robotic environment, the rendering updates are performed manually using Python code. However, this does not alter the efficacy of our experiments.

At the start of each episode, the environment is initialized using a non-zero number of rovers, and a corresponding number of goals. For ease of understanding, the rovers and their goal locations are drawn using the same colours. Sand can be drawn or cleared dynamically using mouse-clicks anywhere within the environment at any time during an episode, allowing us to experiment with changing scenarios online.

Fig 4 shows a screenshot of the system environment and its various components.

B. *The Agents*

Our system utilizes a multiple collaborative Deep Q-Networks (DQN) in a stacked formation, namely, Rover Agents and Cluster Agent. In Mission Scenario 1, only the Rover Agents are used, whereas in Mission Scenario 2, the Cluster Agent is used to control the behaviour of the Rover Agents.

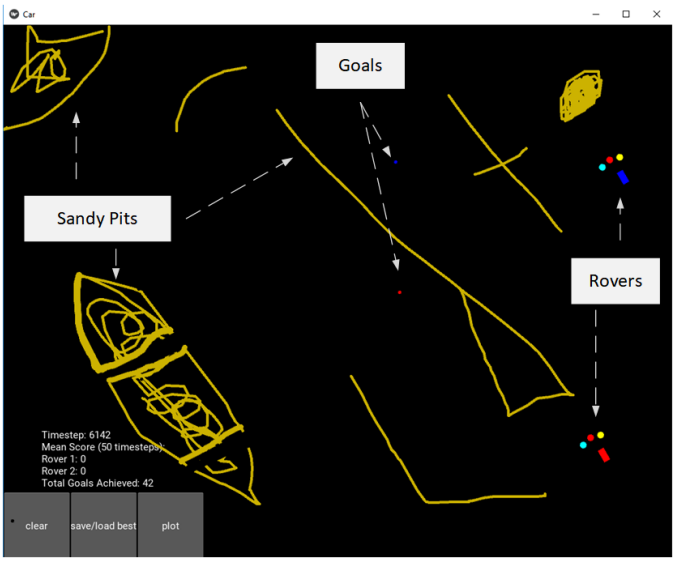

Fig. 4: System Environment

1. *Rover Agent*

Each rover in the environment houses a Rover Agent DQN, which performs the task of controling the rover using localized sensors. It performs this by observing the state of the environment (using ob-board sensors), and performing an action that maximizes its Reward.

In Mission Scenario 1, the Rover Agents act independently of each other, as would be the case when the rovers are unable to communicate with each other.

In Mission Scenario 2, the Rover Agents are able to share information with each other, and thus, collaborate with each other in a manner similar to that of Double Deep Q-Networks. Specifically, each Rover Agent has access to the model parameters of every other Rover Agent, and they use this information to mimic the most "successful" Rover Agent (the one that has the highest average goals-per-timestep).

State:

At each timestep, each Rover Agent receives the following state information from its sensors:

- Its own location within the enviroment at current timestep, t, $(x_{r(t)}, y_{r(t)})$
- Sensor 1 Data: Forward looking
- Sensor 2 Data: 30 degrees to the Left
- Sensor 3 Data: 30 degrees to the Right

Actions:

At each timestep, each Rover Agent is able to perform the following Actions:

- Continue on course
- Rotate 20 degrees Right
- Rotate 20 degrees Left

Rewards:
The Rewards each Rover Agent receives are based on its performance. The Rover Agent may receive these rewards either directly from the environment (Mission Scenario 1), or from the Cluster Agent (Mission Scenario 2). More specifically, in Mission Scenario 1, the rewards received are manually tuned for optimal performance (by the researcher), whereas in Mission Scenario 2, the rewards are tuned by a second DQN.

Each Rover Agent receives the following rewards:

- Living Penalty: A negative reward at each timestep that the rover has not achieved its goal, $(x_{r(t)}, y_{r(t)}) \neq (x_g, y_g)$
- Getting Closer Reward: A small positive reward (less than the living penalty) at each timestep when the rover has gotten closer to the goal, $|(x_{r(t)}, y_{r(t)}) - (x_g, y_g)| < |(x_{r(t-1)}, y_{r(t-1)}) - (x_g, y_g)|$
- Collision Penalty: A large negative reward whenever two rovers collide, $(x_{r,a}, y_{r,a}) = (x_{r,b}, y_{r,b})$
- Sand Penalty: A negative reward whenever the rover enters a sandy pit.
- Goal Reached Award: A large positive reward whenever the rover reaches its goal, $(x_{r(t)}, y_{r(t)}) = (x_g, y_g)$

Fig 5 shows the architecture of the Rover Agent neural network. The architecture is simple by design, since the relationship between the input to output is, although non-linear, not very complex.

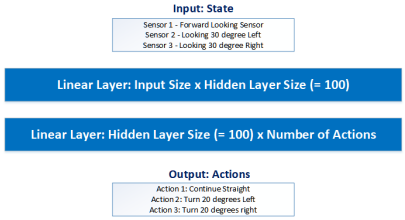

Fig. 5: Rover Agent Architecture

2. Cluster Agent

The Cluster Agent is used only in Mission Scenario 2. It receives the state and cumulative rewards from each Rover Agent as input. It monitors the behaviour of all Rover Agents within the environment using this input, and modifies the Rewards they receive in order to control their behaviour. In other words, the "Actions" performed by the Cluster Agent is the tuning of the Rover Agent's Rewards.

State:
At each timestep, the Cluster Agent receives the following state information from each rover and the environment:

- Location and orientation of each rover within the enviroment at current timestep, $(x_{ra(t)}, y_{ra(t)}, o_{ra(t)}), (x_{rb(t)}, y_{rb(t)}, o_{rb(t)}), ...$
- Location of the goals for each rover, $(x_{ga}, y_{ga}), (x_{gb}, y_{gb}), ...$
- The rewards received by each rover directly from the Environment.

Actions:
The Cluster Agent tunes the rewards each Rover Agent receives for performing their actions, in order to maximize the efficiency of the overall cluster of rovers. At each timestep, it tunes the following rewards:

- Living Penalty for each Rover Agent
- Getting Closer Reward for each Rover Agent
- Collision Penalty for each Rover Agent
- Sand Penalty for each Rover Agent

Rewards:
The Rewards the Cluster Agent receives are based on the cumulative performance of all Rover Agents. The Cluster Agent receives its rewards directly from the Environment:

- Living Penalty: A negative reward at each timestep that the all of the rovers have not achieved their goals
- Goal Reached Award: A positive reward whenever any of the rover reaches its goal

Fig 6 shows the architecture of the Cluster Agent neural network. The architecture is more complicated than that for Rover Agents because the learning task of the Cluster Agent is more complex, i.e., the relationship between the input and output is highly non-linear. The Cluster Agent must evaluate the performance of each Rover Agent in realtime, and then tune its rewards in order to manipulate its future behaviour, all this while the Rover Agent itself is learning how to behave within the Environment.

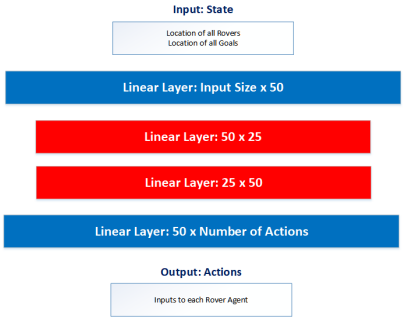

Fig. 6: Cluster Agent Architecture

Episodes, Memory Replay, and Sampling
Since this is a continuous control problem within an enclosed Environment, there is no specific length for an Episode. In each episode, we constantly evaluate the performance of each agent, and store the best performing parameters in memory. For Mission Scenario 2, we update the parameters for every Rover Agent with that of the best, or most successful, Rover Agent (similar to Double Deep Q Networks).

For each DQN within the Environment, we keep track of the previous $100,000$ timesteps in memory. For sampling, we randomly select 100 samples from memory in order to select actions and update the Q-Network.

VI. Performance Evaluation

Performance is evaluated by counting the average number of goals reached and collisions detected by all rovers with respect to time in each episode. Specifically, we monitor the following two metrics:

$$S_T = \frac{\sum_{t=0}^{T} \sum_{r=1}^{k} \Lambda_r}{T} \tag{1}$$

$$C_T = \frac{\sum_{t=0}^{T} \Gamma_r}{T} \tag{2}$$

where:
S_T = Mean Score, i.e., average number of goals achieved by all rovers up to timestep T
C_T = Mean number of collisions achieved up to timestep T
t = Current timestep, ranges from $t = 0$ to rover $t = T$
r = Rover, ranges from Rover 1 to rover k
Λ_r = Goals reached by rover r at timestep t
Γ = Total number of collisions between any two rovers

This simple goal-based evaluation technique allows us to compare the performance of the system independent of the Mission Scenario.

VII. Experiments, Results and Discussions

A. *Mission Scenario 1 - Results and Discussion*

We ran Mission Scenario 1 for a case with 3 Rovers for multiple episodes, and present the results from one of those episodes. The plots for Mean Score and Mean Collisions are shown in Fig 7. A screenshot of the Environment taken at a random timestep within the episode is shown in Fig 8.

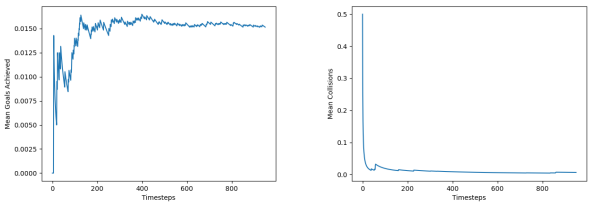

Fig. 7: Mean Score and Collisions for Mission Scenario 1

It can be seen that, when the rewards are manually tuned, the Rover Agents learn how to achieve their goals quickly while avoiding collisions. From the plots it can be seen that at the end of 900 timesteps, the Mean Score is 0.015 goals per timestep, indicating that we achieve a goal, on average, once every 67 timesteps (0.015^{-1}). Similarly, our Mean Collision score is nearly 0, indicating that collisions are very rare, despite there being significant blockages due to sandy pits. Because Rover Agent networks are relatively simple, executing 900 timesteps takes only a few minutes.

It takes a lot of effort, however, to tune the rewards manually to achieve this optimal performance, and this is precisely why Mission Scenario 2 is necessary. In real-world dynamic applications, such as planetary robotics, manually tuning rewards for each new Environment is not always possible or desirable.

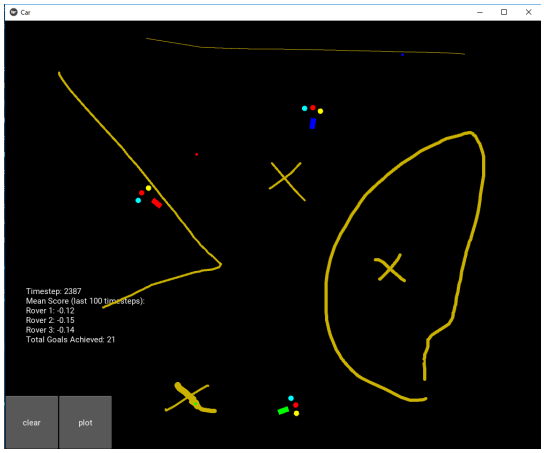

Fig. 8: Mission Scenario 1 - Screenshot

B. *Mission Scenario 2 - Results and Discussion*

We ran Mission Scenario 2 for a case with 2 Rovers for multiple episodes, and present the results from one of those episodes. The plots for the average number of Goals Achieved for one episode of Mission Scenario 2 are shown in Fig 9. A screenshot of the Environment taken at a random timestep within the episode is shown in Fig 10.

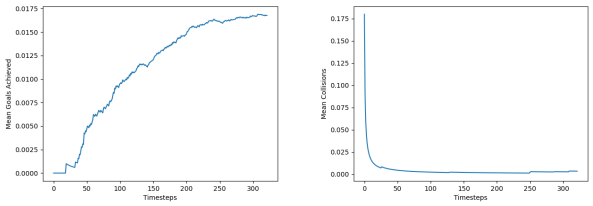

Fig. 9: Mean Score and Collisions for Mission Scenario 2

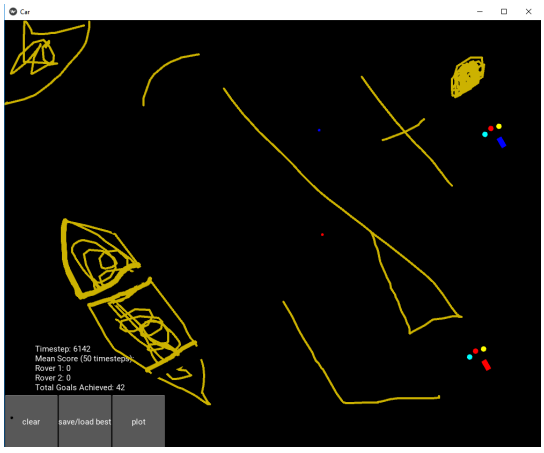

Fig. 10: Mission Scenario 2 - Screenshot

For the chosen episode, the Mean Score is 0.0165 goals per timestep by the end of the episode, indicating that we achieve a goal, on average, once every 60 timesteps (0.0165^{-1}).

Similarly, our Mean Collision score falls to nearly 0, indicating that collisions are very rare, despite there being significant blockages due to sandy pits. Because the complexity of the stacked Cluster-Rover network configuration is greater than just Rover Agent by itself, executing each timestep takes slightly longer.

C. *Comparison - Mission Scenario 1 and Mission Scenario 2*

It can be seen that the combined Rover-Cluster Agents learning (Mission Scenario 2) is slower than that for Rover Agent Only (Mission Scenario 1), which is expected, not only because the Cluster Agent is more complex than the Rover Agent, but also because the stacked configuration adds significant overall complexity. It should be noted that the Cluster Agent isn't controlling the Rover Agents *directly*, but rather, is trying to find ways to *optimize* the overall efficiency of the entire cluster of rovers by auto-tuning the rewards received by the Rover Agents. This is a highly complex task which the Cluster Agent learns to perform fairly well.

Despite the slow learning rate, eventually, the Cluster-Rover Agent configuration of Mission Scenario 2 exceeds the Rover Agent Only architecture of Mission Scenario 1 in terms of overall score, despite running only 2 Rovers instead of 3. This is because:

1) The Cluster Agent automatically tunes the Rewards to each Agent in order to optimize its performance based on its location with respect to the goal location. This is akin to a human being monitoring the behaviour of the entire cluster of rovers, and tuning each rover in order to maximize its performance.
2) We benefit from the fact that individual Rover Agents are able to pass information to each other. Whenever one rover learns a "better" way of achieving its goals, we simply copy its model parameters to all other, less successful, agents. In this way, our system resembles the Double Deep Q-Network system, but instead of having dedicated "prime" and "target" agents, we instead have several "prime" agents working together collaboratively.

Thus, despite its higher complexity, we can see that the Cluster-Rover Agent system is able to learn how to navigate to the goals quite effectively, while avoiding collisions and sandy pits. This is a promising result, one which can be studied in greater detail in future research.

VIII. Conclusion and Future Work

In this paper, we have presented a *TRL 3 - Proof of Concept* for a multi-Rover Guidance and Navigation System using a Stack of Collaborative Deep Q-Networks, and evaluated it on two realistic Mission Scenarios;

1) Where the Rovers' DQNs, called Rover Agents, act independently in an environment with simulated hazards.
2) Where the Rover Agents collaborate with each other, and are guided by a parent DQN called the Cluster Agent. The Cluster Agent continually monitors the behaviour of each Rover Agent, and manipulates their rewards in order to optimize the efficiency of the overall cluster of rovers.

The rovers are equipped with only rudimentary sensors which provide information about their immediate vicinity, and do not have access to global data.

We find that, despite their simplicity, both architectures perform well terms of optimizing time-efficiency. In particular, the Cluster-Rover Agent configuration of Mission Scenario 2 performs quite well in obtaining a high rate of success (obtaining goals), while minimizing risk (collisions).

Although the preliminary results are promising, future work on this topic must look at the following:

- Compare the performance of the developed system with current Industry and Reserach state-of-the-art, especially within the area of Reinforcement Learning.
- Test the system using dedicated dynamic robotics environments, such as Player/Stage, Gazebo, Unity 3D, or MuJoCo.
- Increase the complexity of the mission scenarios by adding realistic conditions such as communication dropouts, component failures, noisy data and sensors, etc.

A. *Code and Implementation*

The code for this project can be found on the author's GitHub page (Github.com/rohaan-ahmed) [5].

References

[1] Exploring the Planets, Smithsonian Natioanl Air and Space Museum. Accessed: Apr 2021. https://airandspace.si.edu/exhibitions/exploring-planets

[2] Pete Sullivan, Mars Perseverance Rover and the Future Colonization of Mars. Published: Mar 2021. Accessed: Apr 2021. https://www.planetizen.com/blogs/112375-mars-perseverance-rover-and-future-coloniz

[3] Mars Exploration Rovers, NASA. Accessed: Apr 2021. https://mars.nasa.gov/mer/mission/rover-status#opportunity

[4] Technology Readiness Levels (TRL), European Space Agency. Accessed: Apr 2021. https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Shaping_the_Future/Technology_Readiness_Levels_TRL#:~:text=Technology%20Readiness%20Levels%20(TRL)%20are,maturity%20level%20of%20a%20technology.&text=ESA%20uses%20the%20ISO%2016290,across%20a%20range%20of%20TRLs

[5] Multi-Rover Guidance and Navigation using Collaborative Deep Q Networks, Rohaan Ahmed (Github). https://github.com/rohaan-ahmed/Master-Repository/tree/master/Multi-Rover%20Guidance%20and%20Navigation%20using%20Collaborative%20Deep%20Q%20Networks