Project Title

……………………..,

…………………………….

Date: 11/26/2022

Supervisor: Group Members:

Dr Farhan Khan Jahanzaib Khan Ludin

Faiz ul Hasan Gardezi

Zain ul Furqan Shahid

Awaiz Adnan

Co-Supervisor:

Engr Badre Munir

**Revision History:**

|  |  |  |
| --- | --- | --- |
| ***Revision History*** | ***Date*** | ***Comments*** |
| 1.00 |  |  |
| 2.00 |  |  |
|  |  |  |

**Document Approval:**

The following document has been accepted and approved by the following:

|  |  |  |
| --- | --- | --- |
| ***Signature*** | ***Date*** | ***Name*** |
|  |  |  |
|  |  |  |
|  |  |  | |

List of Contents

[1 INTRODUCTION 5](#_Toc120386687)

[1.1. PURPOSE 5](#_Toc120386688)

[1.2. PRODUCT SCOPE 5](#_Toc120386689)

[2 OVERVIEW 6](#_Toc120386690)

[2.1 THE OVERALL DESCRIPTION 6](#_Toc120386691)

[2.2 PRODUCT PERSPECTIVE 6](#_Toc120386692)

[2.2. PRODUCT FUNCTIONS 6](#_Toc120386693)

[2.3. USER CHARACTERISTICS 6](#_Toc120386694)

[2.3. CONSTRAINTS 6](#_Toc120386695)

[2.4. ASSUMPTIONS AND DEPENDENCIES 6](#_Toc120386696)

[3 STATE OF THE ART 7](#_Toc120386697)

[3.1 LITERATURE REVIEW 7](#_Toc120386698)

[3.2 EXISTING SYSTEMS 7](#_Toc120386699)

[4 USER/SYSTEM REQUIREMENTS 7](#_Toc120386700)

[4.1 External Interface Requirements 8](#_Toc120386701)

[4.1.1 User Interfaces 8](#_Toc120386702)

[4.1.2 Hardware Interfaces 8](#_Toc120386703)

[4.1.3 Software Interfaces 8](#_Toc120386704)

[4.1.4 Communication Interfaces 8](#_Toc120386705)

[5 Functional Requirements 8](#_Toc120386706)

[5.1 Functional Requirements with Traceability information 8](#_Toc120386707)

[6 Nonfunctional Requirements & Software System Attributes 9](#_Toc120386708)

[6.1 Performance Requirements 9](#_Toc120386709)

[7 Project Design/Architecture 9](#_Toc120386710)

List of Figures

[Figure 1 Student use cases 24](#_Toc381361180)

[Figure 2 Faculty member's use cases 25](#_Toc381361181)

[Figure 3 Administrator Use cases 25](#_Toc381361182)

**List of Tables**

# INTRODUCTION

Scientists have been sending unmanned probes and rovers into the solar system for decades. With a broad number of purposes and targets, these unmanned vehicles have sent back a stunning array of scientific data.

However, there is a critical issue for unmanned, semi-autonomous spaceships: navigating unknown and potentially hazardous terrain. Whilst a human operator can respond to unexpected terrain with their senses, an unmanned, robotic vehicle cannot rely on human operators, especially at such vast distances. They must make use of their programming and having the software to adapt to unknown terrain is essential.

The ability to explore and map previously unknown areas is key for missions using autonomous vehicles. It can impact time-critical mission parameters (such as fuel depletion) and, ultimately, can be the difference between the success and failure of the mission itself. Deep Reinforcement Learning is a field of computer science that can provide software-based solutions to this mission-critical challenge.

## PURPOSE

The purpose of this project is to explore reinforcement learning and further our research into areas where it is still in infancy, we can use this discovery rover or autonomous rover to discover unknown areas without any prior training. The model is still being trained in real life and able to handle situations a normal deep learning model cannot.

## PRODUCT SCOPE

The scope of this project is to build an autonomous rover that can explore areas initially in our university premises and encounter any obstacle and be able to compute an optimal policy in the virtual simulation and take that decision.

Table 1: Terms used in this document and their description

|  |  |
| --- | --- |
| Name | Description |
| S | Set of environment states |
| A | Available Actions |
| R | Reward |
| at | Action selected at time t |
| st | Environment state at time step t |
|  |  |

# OVERVIEW

## THE OVERALL DESCRIPTION

We are building a system whereby the sensor data is collected and a simulation of the environment is created in real time. Our Reinforcement learning model will then take this data as input tensors and perform iterations with our rover as a simulated agent which in turn gives us the optimal pathway. This pathway is then mapped onto the rover as output. This model will not need pre-trained data or training, hence can work in unknown and inhospitable situations. So, to demonstrate our model capability, we will be building a rover with custom designed electronics and hardware. Primary sensors are LIDAR and SONAR to work in sync to generate the simulation of nearby environment. The rover will have autonomous navigation, obstacle avoidance and geo-locating capabilities.

## PRODUCT FUNCTIONS

Area discovery is the feature that our project or rover will be focused around the basic function will be to find the best path from point A to point B with the help of a virtually simulated 2D environment that will be mapped using LIDAR and proximity sensors then using reinforcement learning algorithms we will find the optimal policy and feed that value back into the rover, all of these computations will be done on a cloud because the micro-controllers are constraint devices and cant handle the load.

## USER CHARACTERISTICS

## CONSTRAINTS

The system is designed to help the user operate the vehicle, and the system should always be operating correctly. If the LIDAR stops working and creating a 2D environment, the system will stop working. This problem is valid until the camera or sound sensors are repaired. Sensors will still be used to obtain distance information. The sensor data will be used with the camera for objects around the vehicle. It requires stable and fast internet connection when locating objects around.

The system must be configured to respond to commands in 500 msec. If there is an obstacle on the road during driving, the response time and brake release time must always be the same to achieve the desired deceleration. In all scenarios, the vehicle speed must be constant.

## ASSUMPTIONS AND DEPENDENCIES

In this project, we have assumed that our rover will work in **total unknown environment** taking the best possible decision in real time. In our system, it is assumed that all traffic signs and the presence of all objects around the vehicle can be clearly and seen. If the vehicle is running, it is assumed that the system is always on and scanned. It is assumed that all system elements are operating properly and there are no abnormal conditions.

Dependencies include Cloud, Web, WiFi, Battery(Power).

# STATE OF THE ART

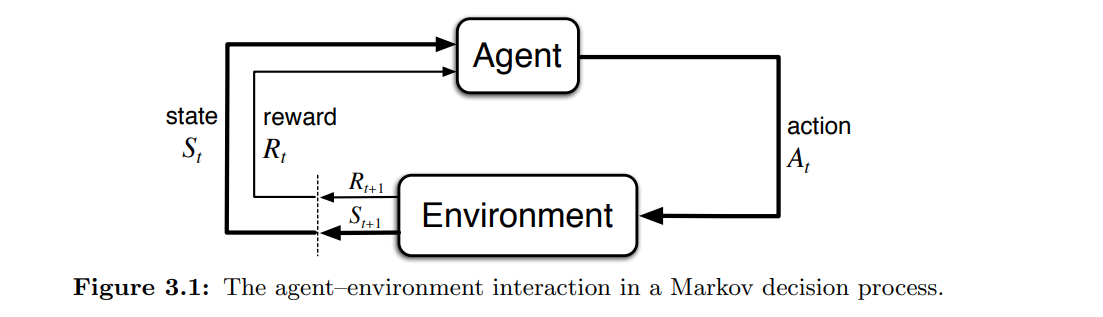
## LITERATURE REVIEW

* **Reinforcement learning**

Reinforcement learning is one of the learning approaches for neural networks. Reinforcement learning implements differently compared to supervised learning and unsupervised learning. Reinforcement learning makes an agent to generate the optimal policy in an environment and maximizes the reward. An agent can learn the rewards given by acting in the respective state and subsequently learn the proper action for each state without having any predefined rules and knowledge about the environment. The agent learns by trials and errors, or in another words, the actions with higher reward are reinforcement while actions with penalties are avoided in the future. It is different from supervised learning because supervised learning requires a huge number of labelled datasets to train the model. The training process is repeated with the same set of data until the model converges. It requires effort in labelling the data and the process is error prone. Reinforcement learning is also different from unsupervised learning because reinforcement learning aims to maximize the reward while unsupervised learning does not.

* **Markov Decision Process**

MDPs are meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The learner and decision maker are called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment. These interact continually, the agent selecting actions and the environment responding to these actions and presenting new situations to the agent.1 The environment also gives rise to rewards, special numerical values that the agent seeks to maximize over time through its choice of actions.



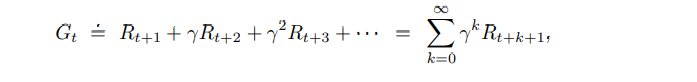
More specifically, the agent and environment interact at each of a sequence of discrete time steps, t = 0, 1, 2, 3.... 2 At each time step t, the agent receives some representation of the environment’s state, St 2 S, and on that basis selects an action, at 2 A(s).3 One time step later, in part because of its action, the agent receives a numerical reward, Rt+1 2 R ⇢ R, and finds itself in a new state, St+1. 4 The MDP and agent together thereby give rise to a sequence or trajectory that begins like this:

S0, A0, R1, S1, A1, R2, S2, A2, R3... (3.1)

In a finite MDP, the sets of states, actions, and rewards (S, A, and R) all have a finite number of elements. In this case, the random variables Rt and St have well defined discrete probability distributions dependent only on the preceding state and action. That is, for particular values of these random variables, s0 2 S and r 2 R, there is a probability of those values occurring at time t, given particular values of the preceding state and action:

p (s0, r|s, a). = Pr {St =s0, Rt =r | St1 =s, At1 =a}, (3.2)

Discount factor γ discounts the rewards of future states contributing to the cumulative reward [2]. The discounted future rewards can be formulated with the formula below.

(3.3)

The value function can estimate the value of a state sequence start with state S, as shown follows:



(3.4)

The optimal value function is the value functions that give the highest value for all states.

(3.5)

* **Q-learning**

Q-learning is a reinforcement learning mechanism that compare the expected utility of available actions given a state. Q-learning can train a model to find the optimal policy in any finite MDP. The Bellman equation suggests the Q-value function as shown in equation (3.6)



(3.6)

The difference of Q-value function in equation 7 with value function in equation 3 is Q-value function estimates the maximum future reward for taking action a given a state s . The calculation of Q-value function takes into account of the maximum discounted future reward for an agent to move from state s(t) to state s(t+1)′. The Q-value function can be iteratively converged to the optimal Q-value function by the difference of the estimated utilities between current state s(t), and next state s(t+1)′,12 with learning rate α, at time step unit t, as shown in equation 3.7.



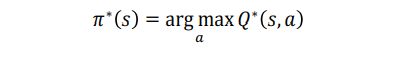
(3.7)

Then, Q-learning function can substitute the value function in equation 3.5, as shown below.

Text

Description automatically generated with low confidence(3.8)

The optimal policy can be retrieved from the optimal value function with equation 3.9.

(3.9)

## EXISTING SYSTEMS

Looking at the literature, there are a couple of works that tried to accomplish these capabilities using methods such as Ant Colony Optimization (ACO) [1], Genetic Algorithm (GA) [2], Particle Swarm Optimization (PSO) [3], Fuzzy Neural Network (FNN) [4], Learning from Demonstration (LfD) [5] and Reinforcement Learning (RL) [6]. However, in this paper we are interested in methods that are based on machine learning methods and specifically RL [7]. For example, Lee et al. [8] trained a network for their RL agent so their quadruped robot could avoid obstacles. In Kominami et al. [9], an RL agent is used in combination with virtual repulsive method by a multi-legged robot to tackle obstacles using its tactile sensors. Zhang et al. [10] used model predictive control (MPC) to train a RL agent to learn to fly a quad-copter; MPC is necessary for training but not necessary for the testing phase. In Zhang et al. [5], the Gaussian mixture model (GMM) and Gaussian mixture regression (GMR) are used for learning from demonstration LfD with the goal of avoiding obstacles. Sadeghi and Levine [11] used deep reinforcement learning (DRL) to make a drone explore and avoid obstacles using a monocular camera. A Deep Q-Network (DQN) [12]-based method was used in [6,13] to train their robot to explore autonomously and avoid obstacles while they used initialized weights for their network (weights are generated using a supervised learning method). Smolyanskiy et al. [14] used off-the-shelf hardware to design a deep neural network (DNN) called TrailNet for following a trail. In [15], we have developed an algorithm that learns from scratch and in an autonomous way using RL and Multi-Layer Perceptron (MLP) to explore autonomously. Some of the aforementioned works are using mono-chrome cameras and use a network in front of the mono-chrome camera to convert the RGB image to a depth image. Some other works use external devices such as Vicon system for the generation of reward and state, which emphasizes the fact that the system needs the external systems for training or even working. Furthermore, there are works that use DNNs, and works that use continuous action space, and works that use a real robot in order to implement their algorithms. Nonetheless, works that use a combination of all the aforementioned advantages are harder to find. Thus, our purpose is to design and implement an algorithm that works with DNNs, without the help of external systems and by using a sensor fusion that provides noise-resistant data for our robot so our algorithm can focus on learning autonomously to explore and avoid obstacles using discrete and continuous actions.

# USER/SYSTEM REQUIREMENTS

* Lidar sensor/s: To help map the environment in a virtual simulation.
* Proximity sensor/s: The blind spots from the Lidar sensor will be detected by the proximity sensors.
* Arduino: This micro-controller is used to capture sensor data which will be converted to our required file format and will be sent over to raspberry pi.
* Raspberry pi (Pico): This micro-controller will further process the data and format it into a tensor. Which will be uploaded ESP.
* ESP32: ESP will send the data to the cloud
* --fill in the data related to the rover--

## External Interface Requirements

### User Interfaces

We will develop a simple web interface where an environment will be created such as a room by the rover and then rover will be displayed in that environment. We will be able to monitor its movement on that window.

### Hardware Interfaces

* RP 2040
* USB to UART converter

### Software Interfaces

* IDEs and tools include Arduino, Pycharm, Mujocco, Carla, Gazebo, Linux terminal, Jupyter notebook, Colab, Tensorflow, AutoCad, MAtlab

### Communication Interfaces

* TCP/IP, HTTP, 6Lowpan and some other communication protocols which will be able to integrate our complete project and communication made easy and fast.

# Functional Requirements

* **Sensor data Modeling**

Model Data from the sensors and make a virtual environment in real time this is one of the key features as our model will be trained on this specific data set.

* **Reinforcement Learning**

Our project utilizes reinforcement learning algorithm to train our rover to find the optimal state value pair or the optimal policy, So the algorithm is one of the key functional requirements.

* **Unmanned Rover**

The suspension of our rover will be rocker bogy since we are accounting for unknown environments so we are considering the rugged terrain and any obstacles that may cause damage. The rover will also have hydraulic pumps on each wheel.

* **Cloud connection**

One of the key requirements will be cloud computing since the microcontrollers are constraint and not well equipped to compute a large amount of data set which will be done through ESP.

* **GPS:** Space-based satellite navigation system that provides time and location information anywhere.
* **Digital Maps:** The process in which data collection is compiled and formatted in a virtual image.
* **Adaptive Cruise Control:** Tracks distances to adjacent vehicles on the same lane. Detects objects in front of a vehicle at risk of emergency collision.

## Functional Requirements with Traceability information

Each requirement will have a separate table

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement ID** |  | | | | **Requirement Type** | | | |  | | | | | **Use Case #** | | | | |  |
| **Status** | ***New*** |  | ***Agreed-to*** | | | | - | ***Baselined*** | | | | - | ***Rejected*** | | | | | - |  |
| **Parent Requirement #** |  | | | | | | | | | | | | | | | | | | |
| **Description** |  | | | | | | | | | | | | | | | | | | |
| **Rationale** |  | | | | | | | | | | | | | | | | | | |
| **Source** |  | | | | | | | | **Source Document** | | | | | | - | | | | |
| **Acceptance/Fit Criteria** |  | | | | | | | | | | | | | | | | | | |
| **Dependencies** |  | | | | | | | | | | | | | | | | | | |
| **Priority** | ***Essential*** | | |  | | ***Conditional*** | | | | - | ***Optional*** | | | | | - |  | | |
| **Change History** |  | | | | | | | | | | | | | | | | | | |

# Nonfunctional Requirements & Software System Attributes

* Performance
* Safety Requirements
* Security Requirements
* Software quality attributes

## Performance Requirements

| **Performance Requirement** | **Description** |
| --- | --- |
| **Response Time** | This system work in real-time. So, response time (T\_response=T\_actuation-T\_event)  must be 500 msecs. |
| **Error Handling** | When an unpredictable failure occurs, system need to recover briefly. |
| **Workload** | System should be able to handle many inputs from its environment in different challenging  weather and traffic conditions. |
| **Scalability** | Sensors and other used hardware tools effective on it but we’re working on simulation  there is no scalability requirement. |

1. **Project Design/Architecture**

* 4+1 ARCHITECTURE VIEW MODEL (mandatory for Software Projects)
  + Use Case View
  + Logical View:
  + Development View
  + Process View
  + Physical View
  + User Interface Design
* Design/flowchart/architecture (Mandatory for research/pure hardware projects)