ML BASED FANET ROUTING

Submitted By

Abhinandan Shah

21MCED12

तमसो मा ज्योतिर्गमय

NIRMA UNIVERSITY

INSTITUTE OF TECHNOLOGY

NAAC ACCREDITED ‘A’ GRADE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2023

ML BASED FANET ROUTING

Major Project-II

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering (Data Science)

Submitted By

Abhinandan Shah

(21MCED12)

Guided By

Dr. Pimal Khanpara

Dr. Pooja Chaturvedi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2023

Certificate

This is to certify that the major project entitled **ML BASED FANET ROUTING** submitted by Abhinandan Shah(Roll No: 21MCED12), towards the partial fulfilment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering (Data Science) of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project part-I, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr. Pimal Khanpara

Dr. Pooja Chaturvedi

Guide & Assistant Professor,

CSE Department,

Institute of Technology,

Nirma University,Ahmedabad

Dr. Swati Jain

Associate Professor,

Coordinator M.Tech - CSE (Data Science),

Institute of Technology,

Nirma University,Ahmedabad

Dr. Madhuri Bhavsar

Professor and Head,

CSE Department,

Institute of Technology,

Nirma University,Ahmedabad

Dr R.N. Patel

Director,

Institute of Technology,

Nirma University,Ahmedabad

Statement of Originality

I, Abhinandan Shah, **21MCED12**, give undertaking that the Major Project entitled **ML-based FANET routing** submitted by me, towards the partial fulfilment of the requirements for the degree of Master of Technology in Computer Science & Engineering (Data Science) of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Date:

Place:

Endorsed by

Prof. Pimal Khanpara

Prof. Pooja Chaturvedi

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to Prof. Pimal Khanpara and Prof. Pooja Chaturvedi, Assistant Professor, Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for her valuable guidance and continual encouragement throughout this work. The appreciation and continual support she has imparted has been a great motivation to me in reaching a higher goal. Her guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Madhuri Bhavsar**, Hon'ble Head of Computer Engineering/ Information Technology Department, Institute of Technology, Nirma University, Ahmedabad for her kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr. R.N. Patel**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

Abhinandan Shah
21MCED12.

Abstract

In recent years, consumer Unmanned Aerial Vehicles have become very popular, everyone can buy and fly a drone without previous experience, which raises concern in regards to regulations and public safety. In this paper, we present a novel approach towards enabling safe operation of such vehicles in urban areas. Our method uses geodetically accurate dataset images with Geographical Information System (GIS) data of road networks and buildings provided by Google Maps, to compute a weighted A* shortest path from start to end locations of a mission. Weights represent the potential risk of injuries for individuals in all categories of land-use, i.e. flying over buildings is considered safer than above roads. We enable safe UAV operation in regards to 1- land-use by computing a static global path dependent on environmental structures, and 2- avoiding flying over moving objects such as cars and pedestrians by dynamically optimizing the path locally during the flight. As all input sources are first geo-registered, pixels and GPS coordinates are equivalent, it therefore allows us to generate an automated and user-friendly mission with GPS waypoints readable by consumer drones' autopilots. We simulated 54 missions and show significant improvement in maximizing UAV's standoff distance to moving objects with a quantified safety parameter over 40 times better than the naive straight line navigation.

Contents

Chapter 1

Introduction

UAV's are a becoming more useful and common in military as well as the daily life applications. In the USA, the FAA gives measures to inform hobbyists to follow certain guidelines to promote the safety usage of UAV's. Previous research as in "Autonomous navigation for low-altitude UAVs in urban areas" by Thomas Castelli et al. described a form of navigation in which it would be safer to fly UAV where they defined different safe zones on a map. The goal of that work was creating the shortest path that would go over safe zones most of the time and go over the danger zones with minimal time. In their work, they used A* algorithm [put a citation here] to compute that safest and shortest distance where they used different weights for the different zones on a map.

This independent study aimed to extend that the work of Castelli et al. to study the performance of deep reinforcement learning algorithms on performing the same tasks for UAV path planning. Similar to the approach followed in [1], in this work, we also assume a UAV with video camera flying over a given geographical region, for which both the geodetically accurate reference image of the area and the GIS data of buildings and roads in the area are available. The dataset used in safe UAV flight includes geo-locations from WPAFB, PVLabs, downtown Orlando, FL and UCF. The image used for this test is of downtown Orlando.

Chapter 2

Literature Survey

Many different topics are studied to enhance the usability and to develop new functions to make drones more capable and autonomous. There are several subfields which are related to this work including video geo-registration, detection and tracking of moving objects in videos, detection of roads, buildings, water bodies from satellite imagery and flight path planning. The most popular trend in UAV video analysis has been moving object detection and tracking from aerial images, many approaches have been proposed with or without using GIS data and geo-registration steps.

Kimura et al. use epipolar constraint and flow vector bound to detect moving objects, Teutsch et al. employ explicit segmentation of images, Xiao et al. restrain the search on the road network, and Lin et al. use a motion model in geocoordinates. Moving object detection and tracking are mainly used to follow targets, for surveillance as Quigley et al. and Rafi et al. describe with their flight path adaptation solutions, or for consumer applications at very low-altitude as in and. Another area that has been getting a lot of attention is autonomous navigation.

Different subproblems have been studied, path planning in dynamic environment, GIS-assisted and vision-based localization using either road detection, buildings layout or DEM (Digital Elevation Map). Various methods have been proposed for UAV navigation, using optical flow with or without DEM, or using inertial sensors.

Obstacle avoidance is also a big concern for automating UAV operation, but research has mostly been focused on ground robots [20], [22], even if there has been adaptations for UAVs as Israelsen et al.'s intuitive solution for operators [21]. The approaches for autonomously navigating UAVs have been studied, but previous work focus on target following or keeping the UAV's integrity. However, in this paper we propose an autonomous UAV navigation method in order to increase public safety in regards to drones operation, and also to prevent UAVs finding themselves in difficult situations.

2.1 Techniques

In our approach, we will be tackling the path planning problem similar to the classic grid world example of AI, in which an agent must walk through cells (or in our case pixels) to achieve a goal. We take our input data as a Google API generated image as shown in Fig 1.

We then scan the image and create a 2D array of rewards based on the color of the image as inked in by the Google API. The image has been stitched together from several other images to achieve a high resolution, for our purposes we have resized the image to a resolution of 300 x 300 pixels for simplicity of training.

Figure 1 The architecture of this Deep Reinforcement Learning agent is that of Q learning and using a deep network to approximate the Q values.

We use experience replay as a form of decoupling sequential information from path planning and being able to determine q values of future locations without bias. The input, or observation, of the agent is defined as a o distance from the target normalized with the pixel length of the image. We then train the network from the experienced actions and determine the Q value using the bellman equation as described in [5].

Terminology

D = Replay Memory

N = Memory Capacity

A = action

R = reward

Transition = [state, reward, action, next state]

Equation 3 = Bellman Equation

Method

Fitting this algorithm to our case, we utilize the array of rewards as the environment, for which the Neural Network will determine Q values to store in replay memory. The Q values will be updated during training, we choose Q-values using an epsilon greedy policy where epsilon is 0.9. The list of available actions in our case 8 directional control.

Actions = [UP, DOWN, LEFT, RIGHT, UP-LEFT, UP-RIGHT, DOWN-LEFT, DOWN-RIGHT]

Our Pixel Ratio for these movements have been determined as 1 pixel per movement as the original pixel ratio from [1] is 0.229 m/pixel, given the Velocity V = 5m/s, the minimum pixel movement is that of 1 Pixel. When resized to 300 x 300 the pixel ratio decreases largely, however to ensure movement we bound the lower limit to 1 pixel.

Our means to achieve our goal involves creating a carefully designed Reward function that allows the agent for error but at the same time incentivizes convergence. These values have been determined empirically and have delivered the best results.

We have determined the rewards for each location as follows:

Road Reward: - 1

Building Reward: 1

Neutral Reward: 0

Reward = array[x,y] + (D/(H*U))

H = the height of the image

U = Resolution

D = Negative Euclidean distance from the goal If D < Straight Line distance then D = ed

Chapter 3

Initial tests with no training

In these following experimental results, we run the agent without any training to see the pure randomness that the agent follows.

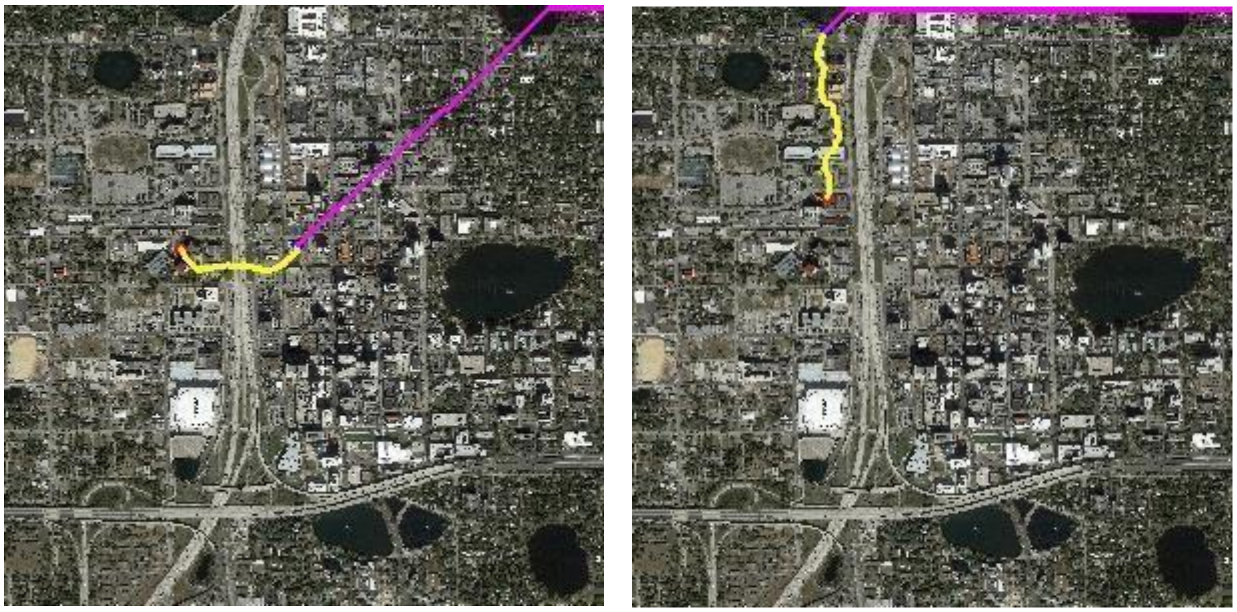


Fig.3 *Figure with different results*

Final tests with present reward function

In these final images, we can see a clear overlap of the agent with the previous algorithm, these two cases by metrics the agent performed better in the static case than the previous algorithm.

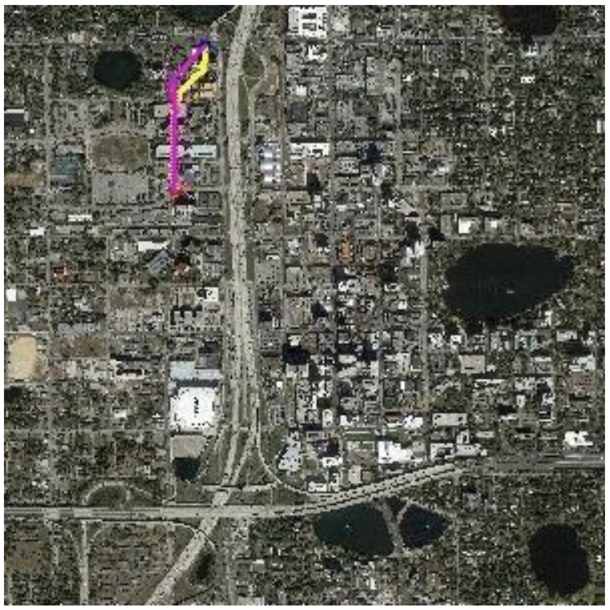
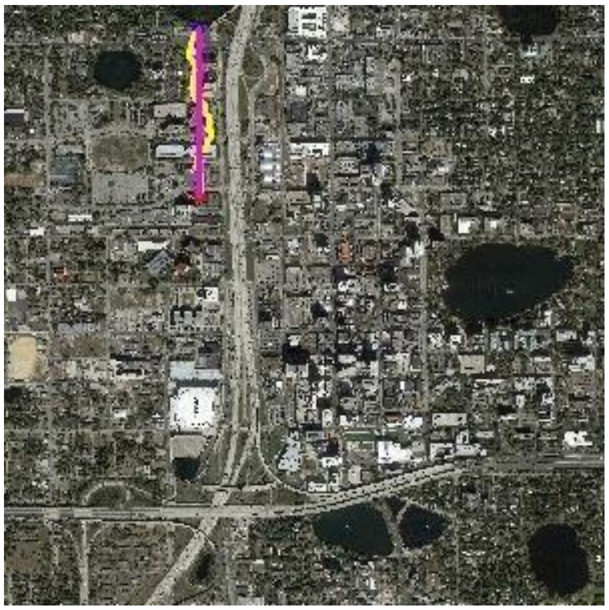
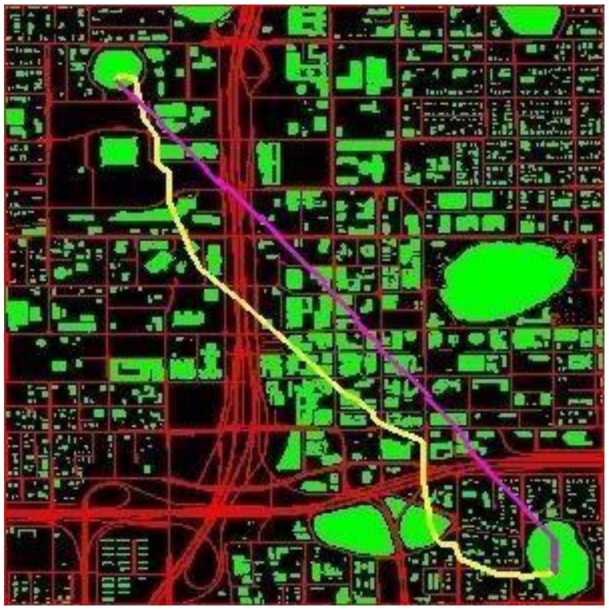
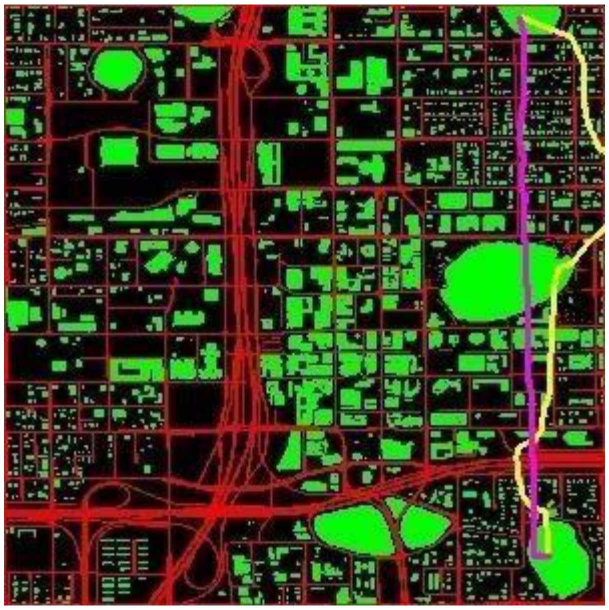



Figure *Figure*

Tests for Long Distances

In these tests, we have colored the map as shown above with the respective reward values given, any areas colored green are considered buildings and have a positive reward of 1, while roads are color coded red and have a negative reward of -1. Transparent areas have no color





Figure

Figure

Chapter 4

Implementation

Fitting this algorithm to our case, we utilize the array of rewards as the environment, for which the Neural Network will determine Q values to store in replay memory. The Q values will be updated during training, we choose Q-values using an epsilon greedy policy where epsilon is 0.9. The list of available actions in our case 8 directional control.

Actions = [UP, DOWN, LEFT, RIGHT, UP-LEFT, UP-RIGHT, DOWN-LEFT, DOWN-RIGHT]

Our Pixel Ratio for these movements have been determined as 1 pixel per movement as the original pixel ratio from [1] is **0.229 m/pixel,** given the Velocity **V** = 5m/s, the minimum pixel movement is that of **1 Pixel.** When resized to 300 x 300 the pixel ratio decreases largely, however to ensure movement we bound the lower limit to 1 pixel.

Our means to achieve our goal involves creating a carefully designed Reward function that allows the agent for error but at the same time incentivizes convergence. These values have been determined empirically and have delivered the best results.

We have determined the rewards for each location

follows:

Road Reward:	- 1
Building Reward:	1
Neutral Reward:	0

Reward = array[x,y] + (D/(H*U))

H = the height of the

image

U = Resolution

D = Negative Euclidean distance from the

goal

If D < Straight Line distance then D $=e^{d}$

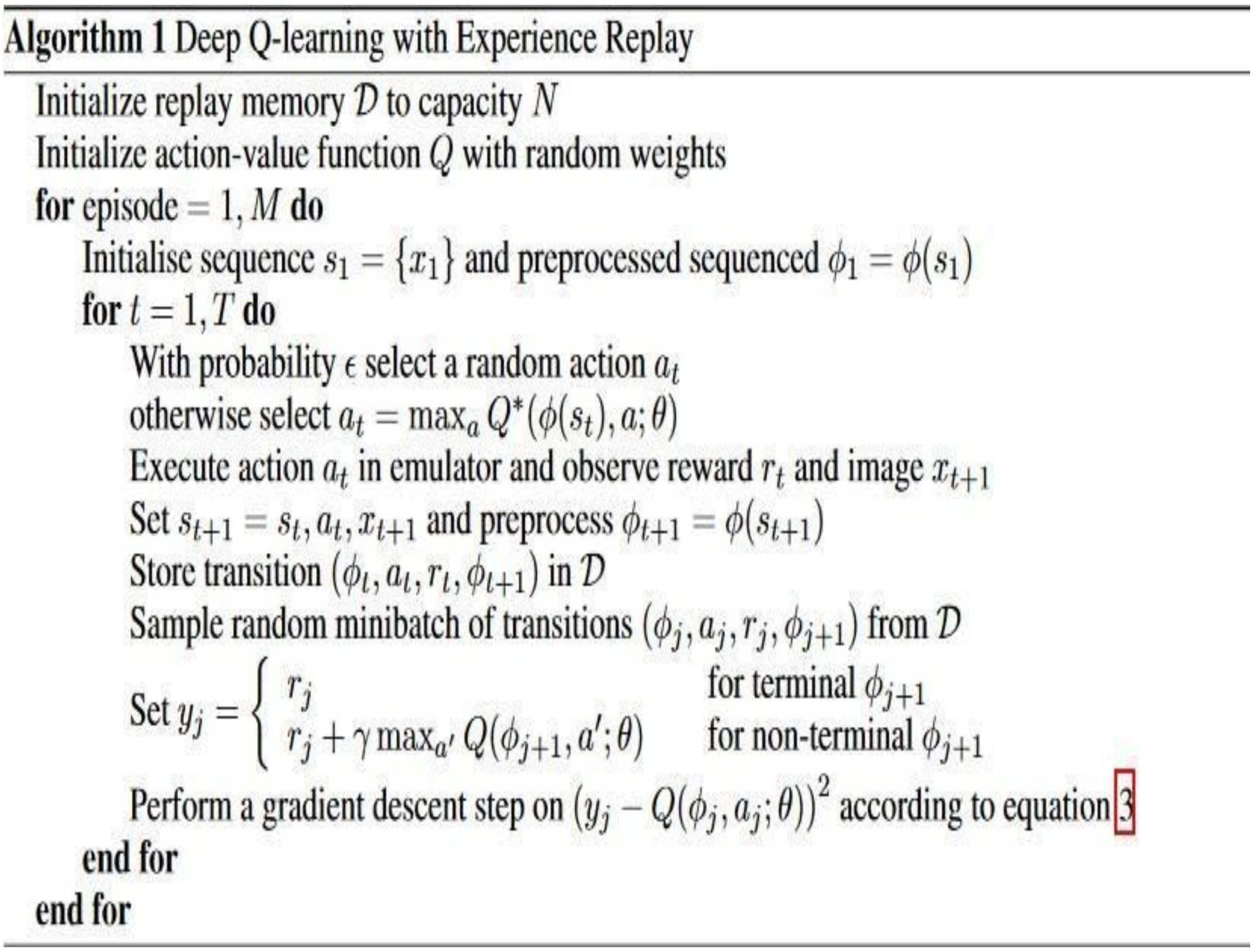
Experiments

In our experimental results we see the evolution of the agent throughout different reward functions. In these following images the A* algorithm's trajectory is presented in yellow, while the DQN is presented in purple.

Initially we explored a plethora of different reward values for buildings in rewards including the original 100, 20, 5 from [1], but as we will see below the current function provided the best results. We also have included results for starting/ending locations with great distances.

Chapter 5

Algorithm



Algorithm 1 Deep Q-learning with Experience Replay

```
Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights
for episode $= 1, M$ do
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ do
        With probability $\epsilon$ select a random action $a_t$
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for
```

$V(s) = \max_a(R(s,a) + \gamma V(s'))$

State(s): current state where the agent is in the environment
Next State(s'): After taking action(a) at state(s) the agent reaches s'
Value(V): Numeric representation of a state which helps the agent to find its path. **V(s)** here means the value of the state s.
Reward(R): treat which the agent gets after performing an action(a).

- **R(s):** reward for being in the state s
- **R(s,a):** reward for being in the state and performing an action a
- **R(s,a,s'):** reward for being in a state s, taking an action a and ending up in s'

e.g. **Good reward** can be **+1**, **Bad reward** can be **-1**, **No reward** can be **0**.
Action(a): set of possible actions that can be taken by the agent in the state(s). e.g. (**LEFT**, **RIGHT**, **UP**, **DOWN**)

Discount factor(γ): determines how much the agent cares about rewards in the distant future relative to those in the immediate future. It has a value **between 0 and 1**. **Lower value** encourages **short–term** rewards while **higher value** promises **lon g-term reward**

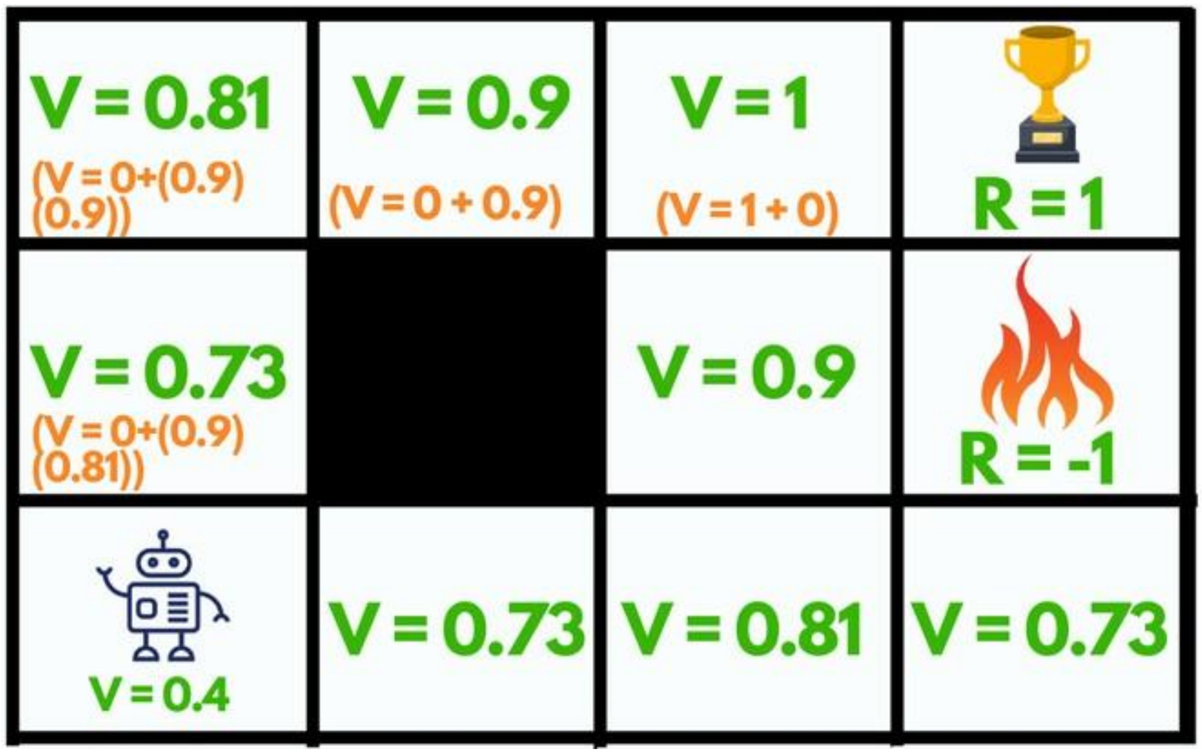


Fig: Using Bellman Equation

The **max** denotes the most **optimum** action among all the actions that the agent can take in a particular state which can lead to the reward after **repeating this process every consecutive step.**

Algorithm 1 Deep Q-learning with Experience Replay

```
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights
For episode = 1,M do
    Initialize sequence s1 = {x1} and preprocessed sequenced Φ1 = Φ(s1)
    For t = 1,T do
        With probability ε select a random action at
        Otherwise select at = MAXaQ*(Φ(st),a;θ )
        Execute action at in emulator and observed reward rt and image xt+1
        Set st+1 = st ,at,xt+1 and preprocess Φt+1 = Φ(st+1)
        Store transition (Φt,at,rt, Φt+1) in D
        Sample random minibatch of transitions (Φj,aj,rj, Φj+1) from D
        Set yj = rj                 for terminal Φj+1      for terminal Φj+1
                rj + γmaxa’Q(Φj+1,a′; θ)      for non-terminal Φj+1
        Perform a gradient descent step on (yj – Q(Φj ,aj; θ))^2
        end for
        end for
```

Chapter 6

Output:-

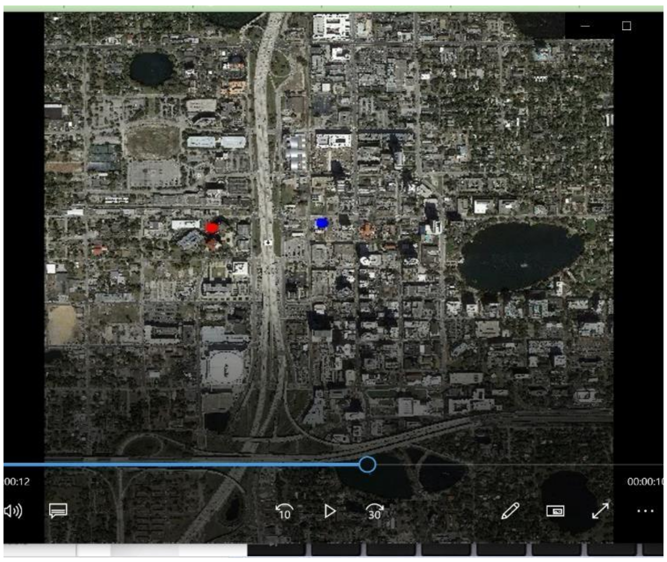


Figure1

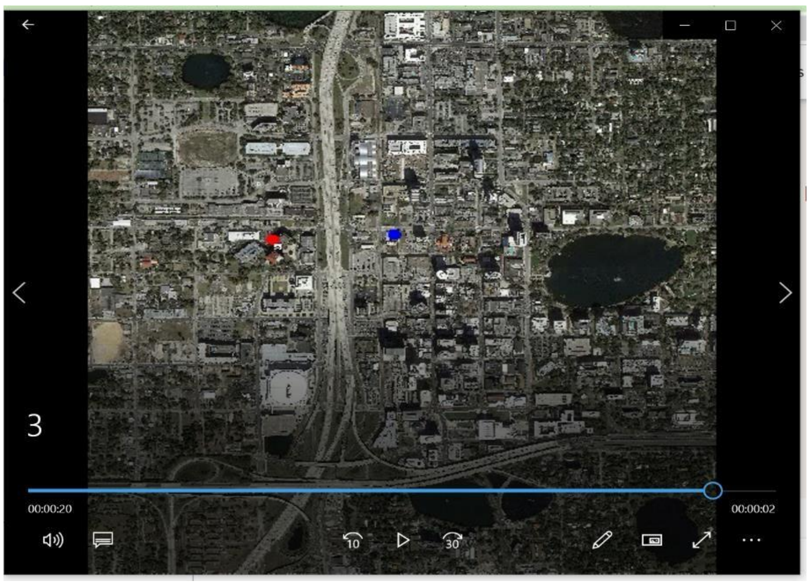


Figure2

Chapter 7

Conclusion

We conclude that DQN offers an alternative solution to the path planning problem that allows for further exploration of traversing a map in a safe way. The advantages to this method is its simple approach allows for easy testing and no need for an internet connection for path planning, just the required input. Not to mention the ever-growing potential of AI and the branch of Deep Reinforcement Learning can improve performance as end-to-end solutions come up. In this independent study we have successfully replicated static path planning for UAVs. Given that dynamically UAVs must always avoid in coming cars in their FOV, traditional dodging path planning can be placed and then the static path reembarked. This is novelty of the approach.

Bibliography

[1] Habbecke et al., ”Automatic registration of oblique aerial images with cadastral maps,” Trends and Topics in Computer Vision, 2010.

[2] Xu et al., ”Real-time 3D navigation for autonomous vision-guided MAVs,” IROS, 2015.

[3] Israelsen et al., ”Automatic collision avoidance for manually teleoperated unmanned aerial vehicles,” ICRA, 2014.

[4] Reinforcement Learning with Tensorflow by Morvan Zhou, https://github.com/MorvanZhou/Reinforcement-learning-with-tensorflow

[5] Playing Atari with Deep Reinforcement Learning by Mnih et al. https://arxiv.org/abs/1312.5602

[6] Q learning a simple proof by Melo, http://users.isr.ist.utl.pt/~mtjspaan/readingGroup/ProofQlearning.pdf

[7] Dueling Network Architectures for Deep Reinforcement Learning By Google, https://arxiv.org/pdf/1511.06581.pdf

[8] AI Safety Gridworlds by DeepMind https://arxiv.org/pdf/1711.09883.pdf

[9] Autonomous navigation for low-altitude UAVs in urban areas by Castelli et al. https://arxiv.org/abs/1602.08141

[10] Cohenour et al., ”Camera models for the wright patterson air force base 2009,” IEEE Aerospace and Electronic Systems Magazine, 2015.

[11] Sheikh et al., ”Geodetic Alignment of Aerial Video Frames,” in Video Registration, Eds. Boston, 2003.

[12] Gonzalez et al., ”Using state dominance for path planning in dynamic environments with moving obstacles,” ICRA, 2012.