

REINFORCEMENT LEARNING BASED AUTOMATED PATH PLANNING IN GARDEN ENVIRONMENT USING DEPTH - 'RapiG-D'

ZEROth REVIEW

- S SATHIYA MURTHI - 2018504604
- PRANAV BALAKRISHNAN - 2018504581
- C ROSHAN ABRAHAM - 2018504591

**BE ELECTRONICS AND COMMUNICATION
ENGINEERING**

**GUIDED BY:
DR. V. SATHIESH KUMAR,
ASSISTANT PROFESSOR,
DEPARTMENT OF ELECTRONICS
ENGINEERING
MIT CAMPUS, ANNA UNIVERSITY**

TABLE OF CONTENTS

- Introduction
- Motivation
- Objective
- Literature Survey
- Methodology
- Current Progress
- Work plan
- References

INTRODUCTION

- Over the years, robots have been implemented in a wide range of applications and environments.
- In turn, this has led to dealing with a host of environments that are increasingly dynamic and unknown.
- Path planning with reinforcement learning would aid the automation of multiple tasks with minimal human intervention.

MOTIVATION

- A learning based approach, can be used to navigate and map an unknown environment and trace the best possible path between any two points.
- Stereo camera based depth mapping would be better suited to identify any type of obstacle.
- Current methods are limited by their inability to adapt to new environments and varying obstacles.

OBJECTIVES

- To develop a path finding algorithm that finds the best possible path between two paths in an unknown garden environment.
- To perform hardware implementation of reinforcement learning based obstacle mapping and analyze its performance.
- To perform detection of plant species using deep learning.
- To develop an algorithm to explore all possible paths and determine the most efficient one.
- To ensure that the robot avoids all obstacles and unnecessary detours.

LITERATURE SURVEY

AUTHOR NAME	TITLE	YEAR OF PUBLICATION	JOURNAL	DESCRIPTION
Qingbiao Li , Fernando Gama , Alejandro Ribeiro , Amanda Prorok	Graph Neural Networks for Decentralized Multi-Robot Path Planning	2020	IEEE/RSJ International Conference on Intelligent Robots and Systems	<ul style="list-style-type: none">• This paper proposed a convolutional neural network (CNN) that extracts adequate features from local observations.• A graph neural network (GNN) that communicates these features among robots.• Model is trained to imitate an expert algorithm and evaluate the method in simulations

LITERATURE SURVEY

AUTHOR NAME	TITLE	YEAR OF PUBLICATION	JOURNAL	DESCRIPTION
Martin Gromniak, Jonas Stenzel	Deep Reinforcement Learning for Mobile Robot Navigation	2019	Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)	<ul style="list-style-type: none">• Developed training procedure, set of actions available, suitable state representation, and a reward function.• Evaluated using a simulated real-time environment• The experimental evaluation showed that DRL can be applied successfully to robot navigation.

LITERATURE SURVEY

AUTHOR NAME	TITLE	YEAR OF PUBLICATION	JOURNAL	DESCRIPTION
Jing Xin, Huan Zhao, Ding Liu	Application of Deep Reinforcement Learning in Mobile Robot Path Planning	2019	IEEE robotics and automation letters	<ul style="list-style-type: none">• A Deep Q-network (DQN) is designed and trained to approximate the mobile robot state-action value function and Q value corresponding to each possible mobile robot action is determined by the well trained DQN.• The current optimal mobile robot action is selected by the action selection strategy to the goal point while avoiding obstacles ultimately

METHODOLOGY

- **Reinforcement learning** is a machine learning training method based on rewarding desired behaviors and/or punishing undesired ones. In general, a reinforcement learning agent is able to perceive and interpret its environment, take actions and learn through trial and error.

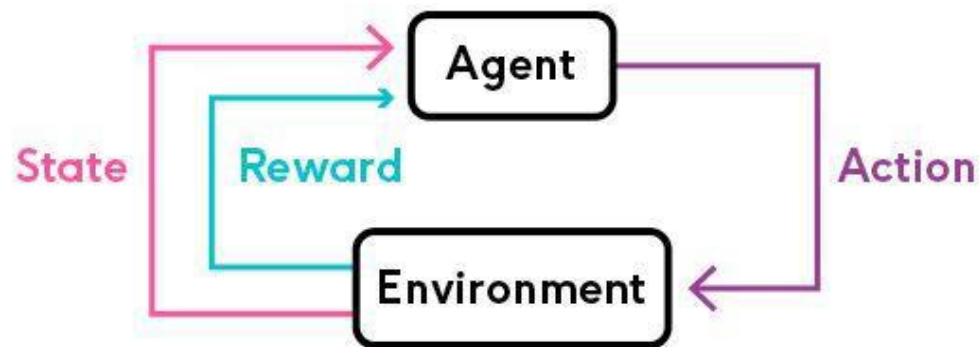


Figure1: SARSA Illustration

Reference: <https://medium.com/@vishnuvijayanpv/what-is-reinforcement-learning-e5dc827c8564>

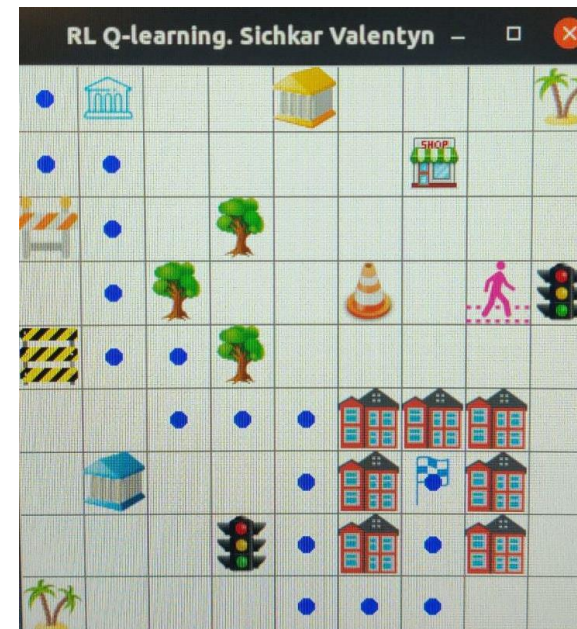


Figure2: SARSA – Software Implementation

METHODOLOGY

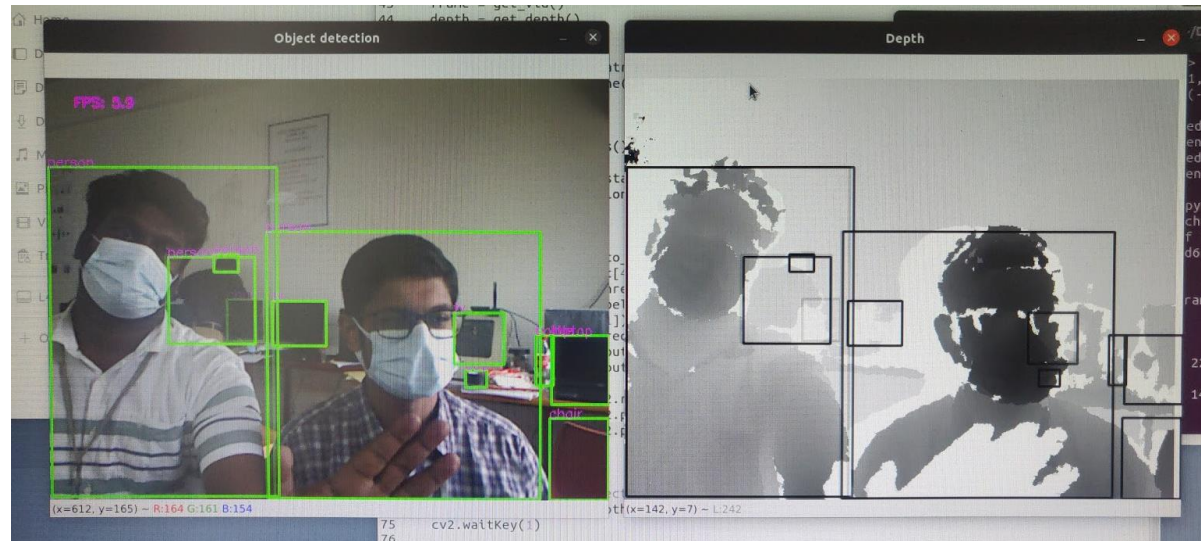
- **Stereo vision** is the computation of depth based on the binocular disparity between the images of an object in left and right eyes
- The **Kinect** camera is a motion sensing input device produced by Microsoft and first released in 2010. The device generally contains RGB cameras, and infrared projectors and detectors that map depth through either structured light or time of flight calculations, which can in turn be used to perform real-time gesture recognition and body skeletal detection, among other capabilities.
- **Image processing** techniques are used to extract and utilize the depth map generated.



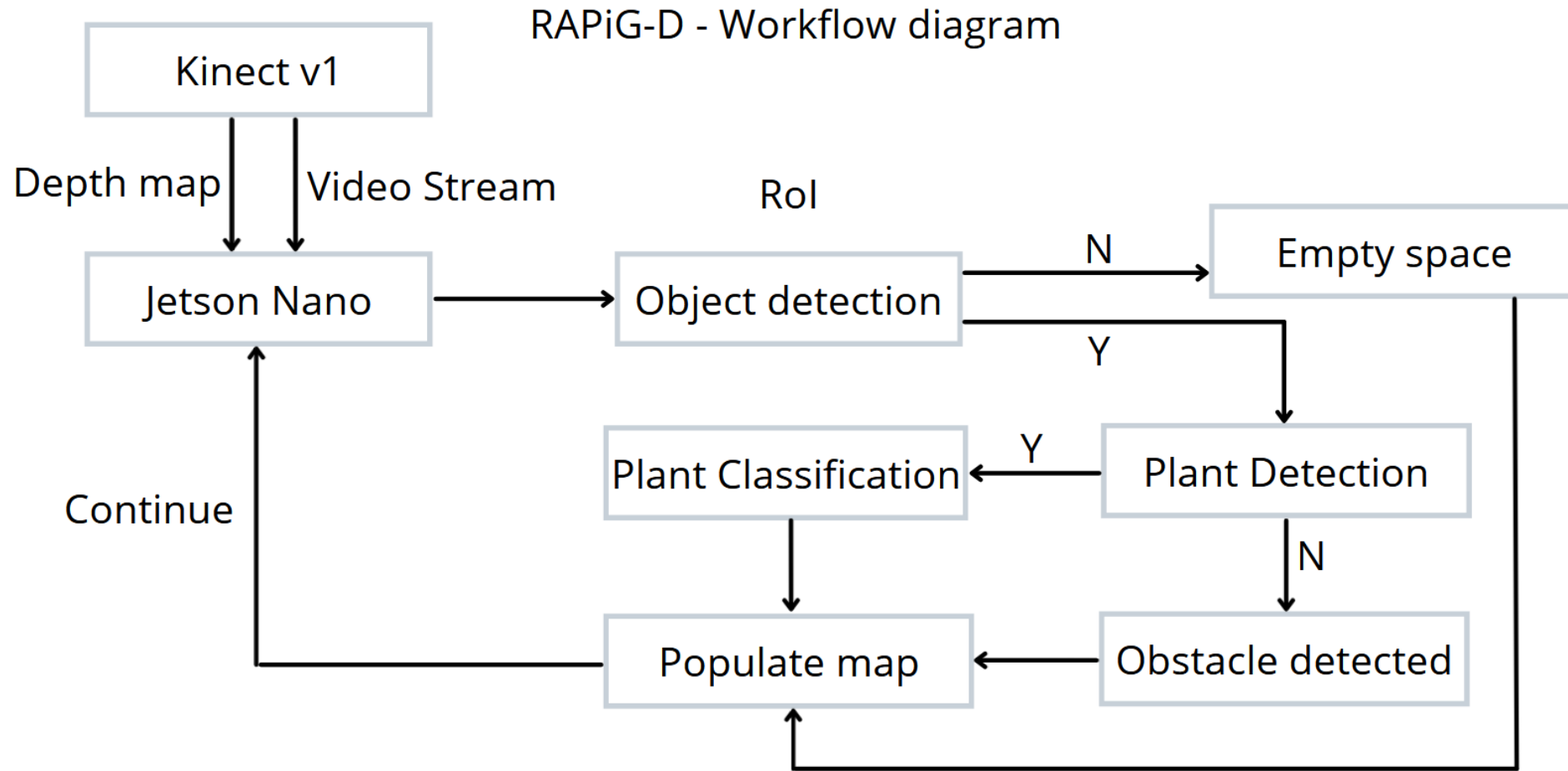
Figure1: XBOX 360 Kinect camera
Reference: <https://en.wikipedia.org/wiki/Kinect>

METHODOLOGY

- Convolutional neural network (**CNN**) is a class of deep learning neural networks. They're most commonly used to analyze visual imagery and are frequently used in image classification.
- CNNs can therefore be used to detect and classify plants after collection of suitable dataset.
- Models like **YOLOv5** can be trained with custom data for the task



METHODOLOGY

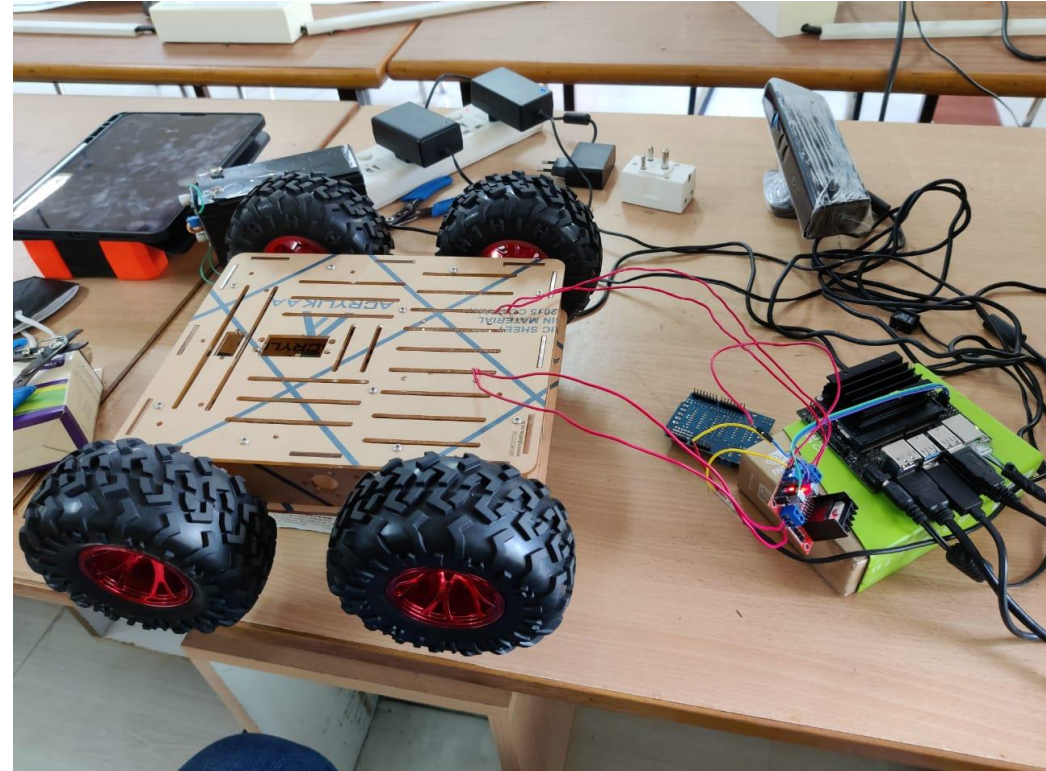


COMPONENTS USED

COMPONENTS	SPECIFICATION	NUMBER USED
Jetson Nano	Developer Kit - 4GB	1
Robot Chassis	Dimensions: 250 x 200 x 46 mm	1
DC motor	200 rpm 12V	2
Wheels	Diameter-130mm Width-60mm	4
Kinect v1 sensor		1
Motor Driver IC	L298N	1
IR sensor	8 channel array	1
Battery	12V7A	1

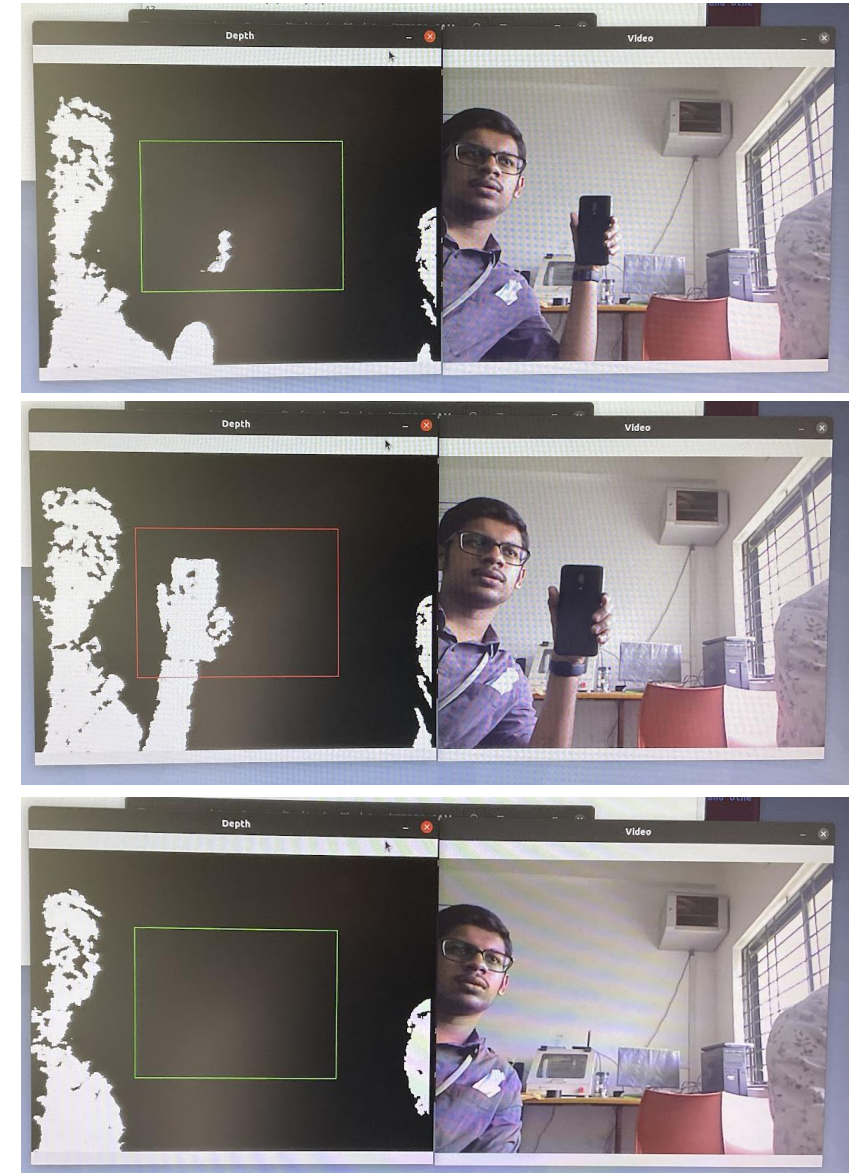
CURRENT PROGRESS

- Tested software implementation of RL SARSA in PC.
- Explored stereo camera and microcontroller options.
- Initialized Jetson NANO and set up bot.



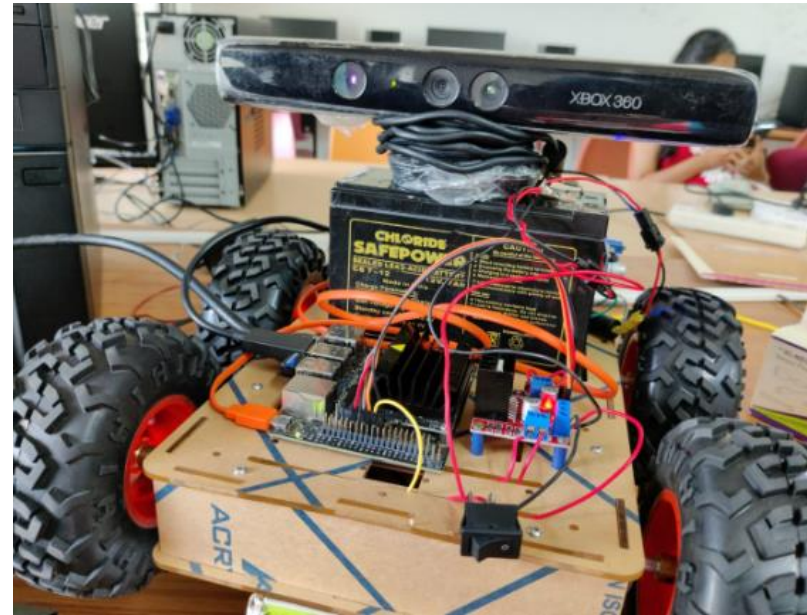
CURRENT PROGRESS

- Interfaced with stereo camera (KINECT v1) on Jetson NANO.
- Obtained depth map and applied depth threshold to identify nearby objects.



CURRENT PROGRESS

- Interfaced with L298N motor driver and enabled four directional movement.
- Set up remote access for Jetson NANO using SSH and VNC.
- Set up suitable power supply for bot portability.



WORK PLAN

First review:

- Set up testing space.
- Study available and suitable RL algorithms.
- Test the performance of the algorithms.

Second Review:

- Collect data set for plant detection model.
- Train plant detection model.

Third review:

- Setup interface for mapping the environment.
- Testing and optimizing.

REFERENCES

- Arga Dwi Pambudi, Trihastuti Agustinah and Rusdhianto Effendi, "Reinforcement Point and Fuzzy Input Design of Fuzzy Q-Learning for Mobile Robot Navigation System", International Conference of Artificial Intelligence and Information Technology (ICAIIIT), September 2019
- Martin Gromniak and Jonas Stenzel, "Deep Reinforcement Learning for Mobile Robot Navigation", Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), July 2019
- Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, T. K. Satish Kumar, Sven Koenig, and Howie Choset, "PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning", IEEE Robotics and Automation Letters, Vol. 4, no. 3, July 2019.
- Hyansu Bae, Gidong Kim, Jonguk Kim, Dianwei Qian and Sukgyu Lee, "Multi-Robot Path Planning Method Using Reinforcement Learning", Multidisciplinary Digital Publishing Institute, Vol 3, No 4, May 2019.
- Jing Xin, Huan Zhao, Ding Liu, "Application of Deep Reinforcement Learning in Mobile Robot Path Planning", IEEE Robotics and Automation Letters, Vol 2, No 3, January 2019.

**END OF
PRESENTATION**

REINFORCEMENT LEARNING BASED AUTOMATED PATH PLANNING IN GARDEN ENVIRONMENT USING DEPTH - 'RAPIG-D'

FIRST REVIEW

- S SATHIYA MURTHI - 2018504604
- PRANAV BALAKRISHNAN - 2018504581
- C ROSHAN ABRAHAM - 2018504591

**BE ELECTRONICS AND COMMUNICATION
ENGINEERING**

**GUIDED BY:
DR. V. SATHIESH KUMAR,
ASSISTANT PROFESSOR,
DEPARTMENT OF ELECTRONICS
ENGINEERING
MIT CAMPUS, ANNA UNIVERSITY**

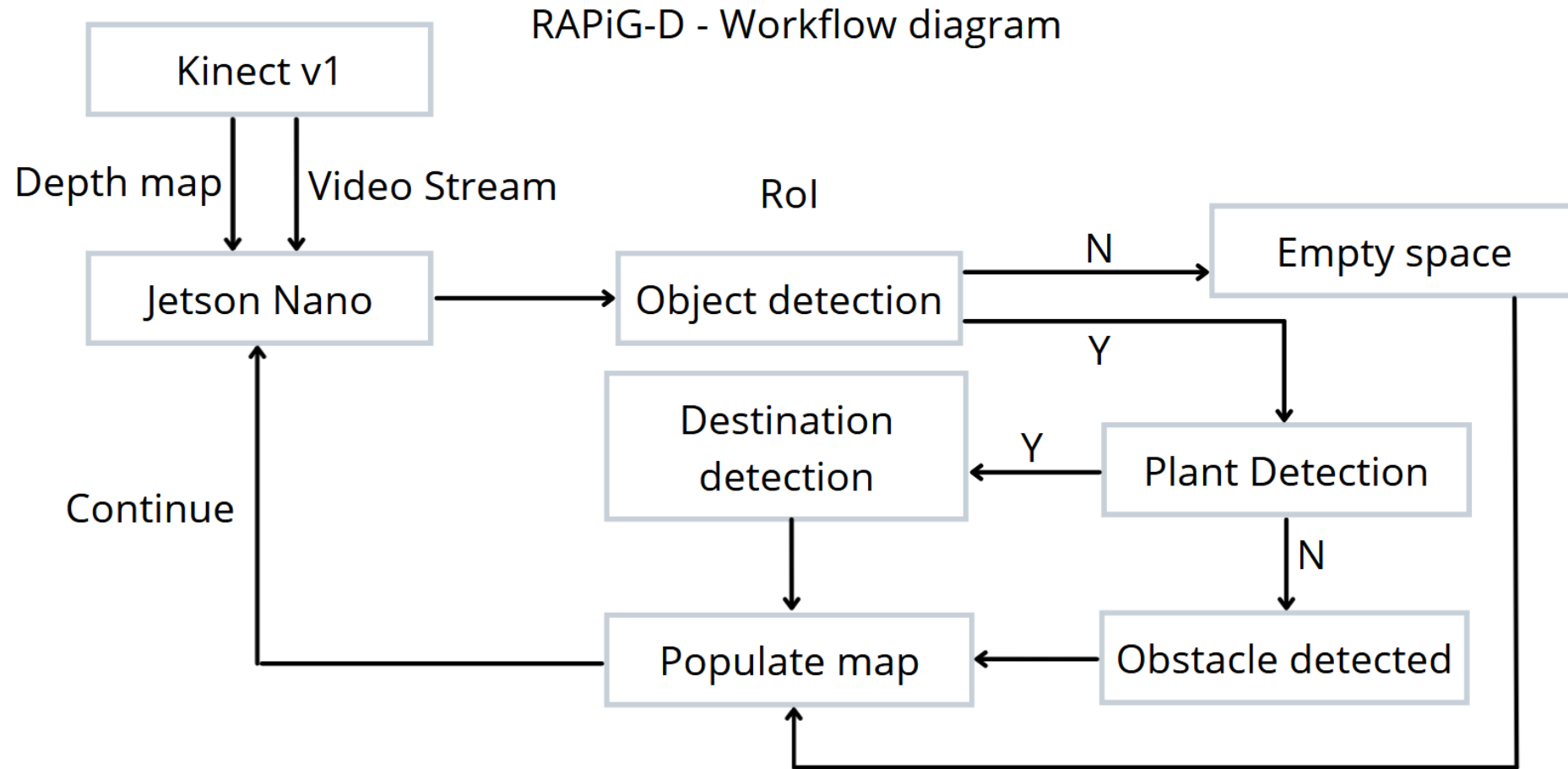
TABLE OF CONTENTS

- Objectives
- Methodology / Block Diagram
- Previous progress Recap
- Bot Components and HW specs
- Current Progress
- Problems Faced
- Work plan
- References

OBJECTIVES

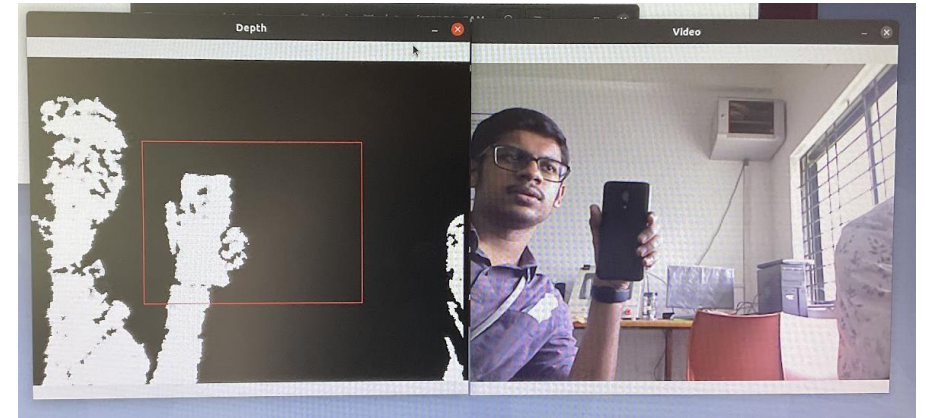
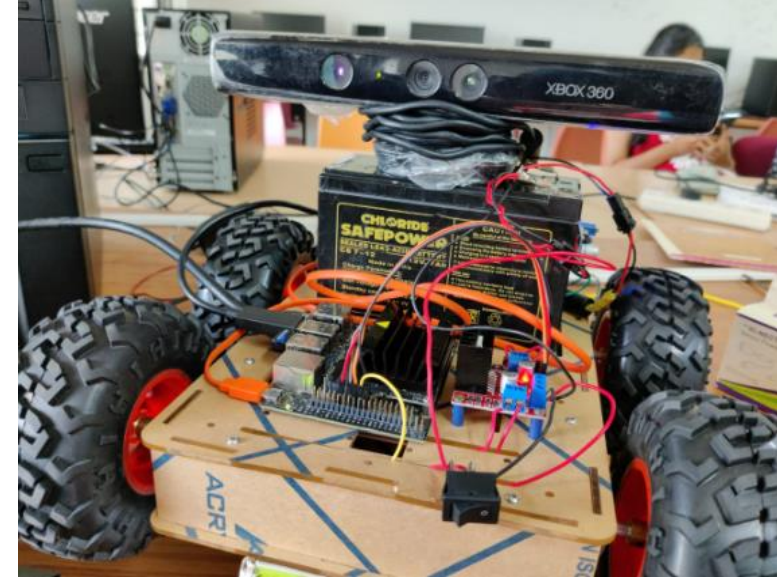
- To perform hardware implementation of reinforcement learning based path planning to generate a map of an unknown environment by employing depth estimation based obstacle detection and analyze its performance.
- To estimate the best possible path to reach a given destination.

METHODOLOGY



PREVIOUS PROGRESS

- Tested software implementation of RL SARSA in PC.
- Initialised Jetson nano and interfaced with stereo camera (KINECT v1) .
- Obtained depth map and applied depth threshold.
- Interfaced with L298N motor driver and enabled four directional movement.
- Set up basic design of robot.

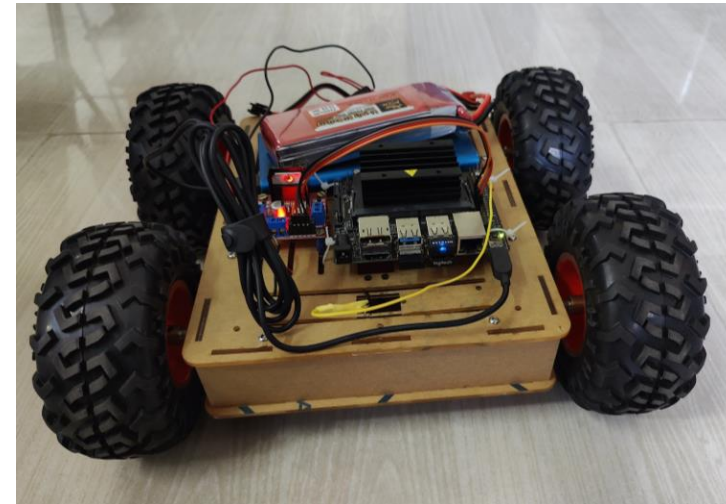


COMPONENTS USED

COMPONENTS	SPECIFICATION	NUMBER USED
Jetson Nano	Developer Kit - 4GB	1
Robot Chassis	Dimensions: 250 x 200 x 46 mm	1
DC motor	200 rpm 12V	2
Wheels	Diameter-130mm Width-60mm	4
Kinect v1 Camera	Stereo camera, IR camera	1
Motor Driver IC	L298N	1
IR sensor	8 channel array	1
Battery	11.1 V 4200mah Li-Po	1

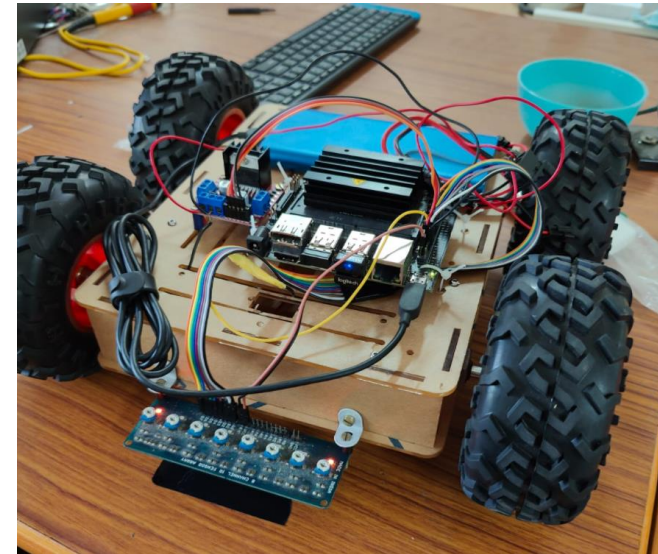
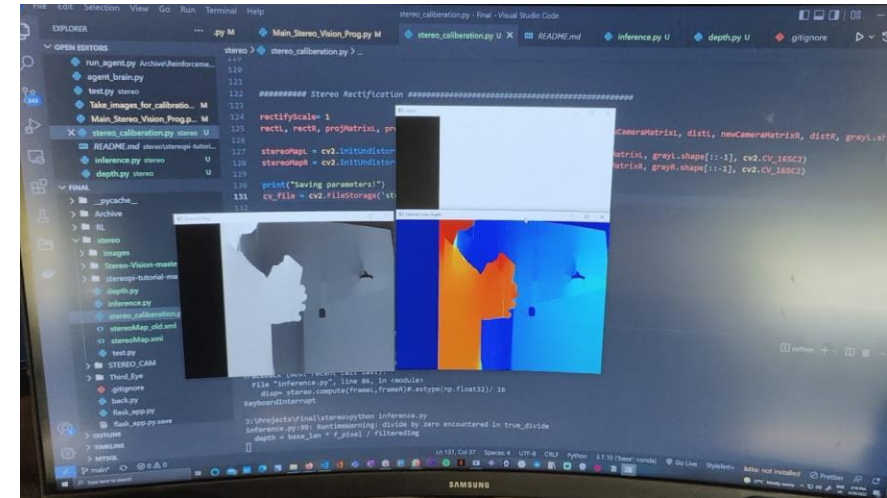
CURRENT PROGRESS

- Tried repositioning Jetson nano and motor driver for better weight distribution.
- Decided to switch to Li-po battery instead of lead acid 12v battery to power the motor driver.
- Set up web output for obstacle detection for remote inference
- Tested mono-camera depth estimation using MIDAS model



CURRENT PROGRESS

- Mounted and used IR sensor array to enable line following for movement calibration.
- Tested speed control of motor by voltage control using DC-DC converter to improve intersection detection
- Implemented stereo camera calibration and depth estimation with two webcams. Performance was not adequate.



CURRENT PROGRESS

- Switched back to Kinect camera
- Tested lane detection for movement calibration.
- Tried to calibrate turn using line following, timing and detecting line crossing
- Optimised object detection to detect obstacles in next cell then integrated it with line following linear movement to set up obstacle mapping in linear space



PROBLEMS FACED

- Bot movement irregular due to weight and loose shaft
- Unable to VNC remotely without display connected. Requires a Dummy HDMI connection.
- Jetson nano and Kinect camera not functioning properly when used in battery mode – 12V 7A Lead Acid battery
- Stability of bot and calibration is an issue.
- Bot doesn't turn due to the weight of the 12V lead acid battery
- Jetson Nano turns off when using Kinect camera when powered by power bank
- Bot turning is irregular – Wheels get stuck, skid. Turning inaccurate.

WORK PLAN

Second Review:

- Set up RL algorithm to map 2D space
- Improve bot stability and calibrate turning

Third review:

- Setup interface for mapping the environment remotely.
- Testing and optimizing.

REFERENCES

- Arga Dwi Pambudi, Trihastuti Agustinah and Rusdhianto Effendi, "Reinforcement Point and Fuzzy Input Design of Fuzzy Q-Learning for Mobile Robot Navigation System", International Conference of Artificial Intelligence and Information Technology (ICAIIIT), September 2019
- Martin Gromniak and Jonas Stenzel , "Deep Reinforcement Learning for Mobile Robot Navigation", Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), July 2019
- Guillaume Sartoretti , Justin Kerr , Yunfei Shi, Glenn Wagner, T. K. Satish Kumar, Sven Koenig, and Howie Choset, "PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning", IEEE Robotics and Automation Letters, , Vol. 4, no. 3, July 2019.
- Hyansu Bae , Gidong Kim , Jonguk Kim , Dianwei Qian 4 and Sukgyu Lee , "Multi-Robot Path Planning Method Using Reinforcement Learning", Multidisciplinary Digital Publishing Institute, Vol 3, No 4, May 2019.
- Jing Xin, Huan Zhao, Ding Liu, "Application of Deep Reinforcement Learning in Mobile Robot Path Planning", IEEE Robotics and Automation Letters, Vol 2, No 3 , January 2019.

**END OF
PRESENTATION**

REINFORCEMENT LEARNING BASED AUTOMATED PATH PLANNING IN GARDEN ENVIRONMENT USING DEPTH - 'RAPIG-D'

SECOND REVIEW

- S SATHIYA MURTHI - 2018504604
- PRANAV BALAKRISHNAN - 2018504581
- C ROSHAN ABRAHAM - 2018504591

**BE ELECTRONICS AND COMMUNICATION
ENGINEERING**

**GUIDED BY:
DR. V. SATHIESH KUMAR,
ASSISTANT PROFESSOR,
DEPARTMENT OF ELECTRONICS
ENGINEERING
MIT CAMPUS, ANNA UNIVERSITY**

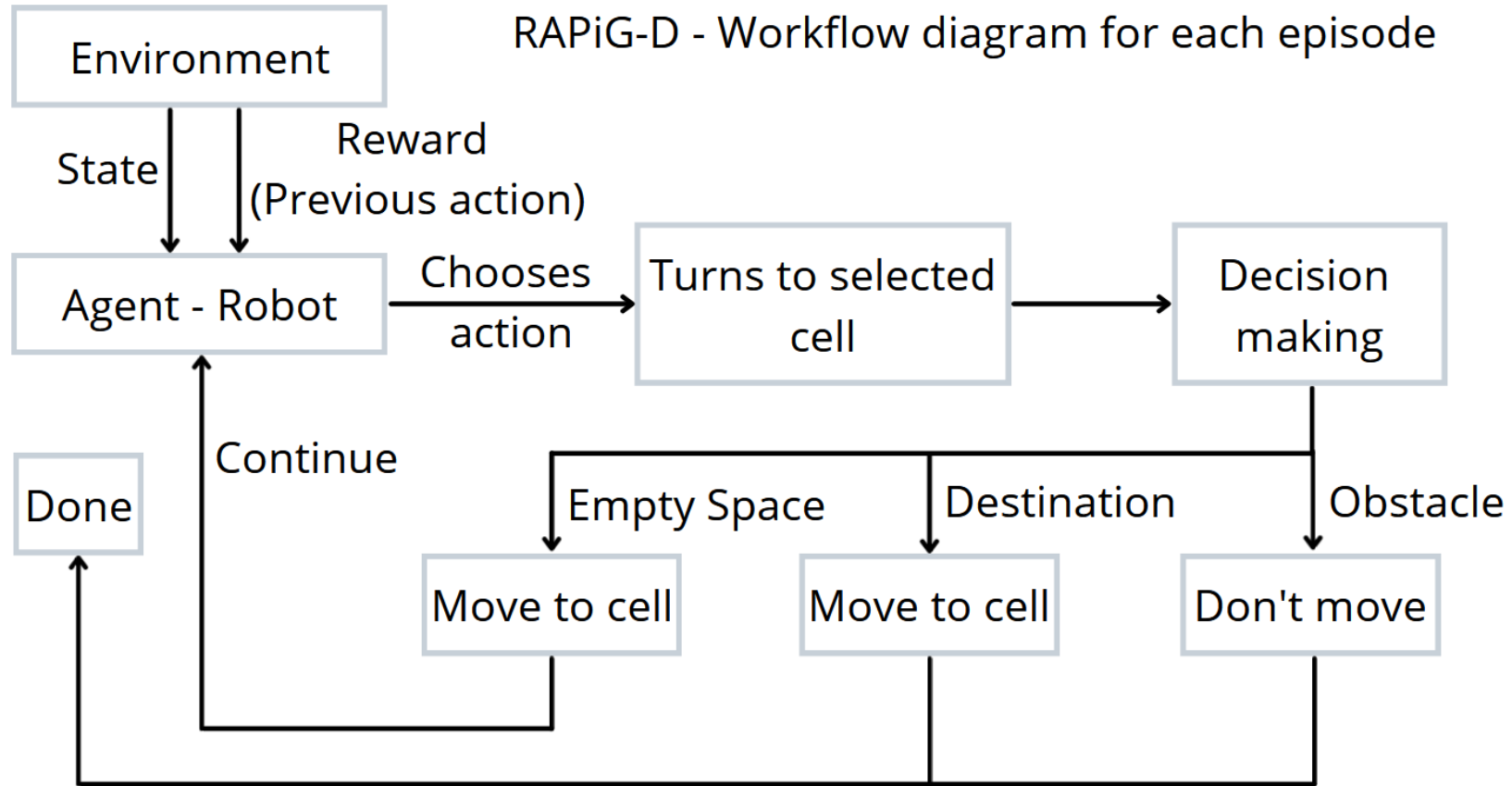
TABLE OF CONTENTS

- Objectives
- Methodology / Block Diagram
- Previous progress Recap
- Current robot design
- Current Progress
- Problems Faced
- Work plan
- References

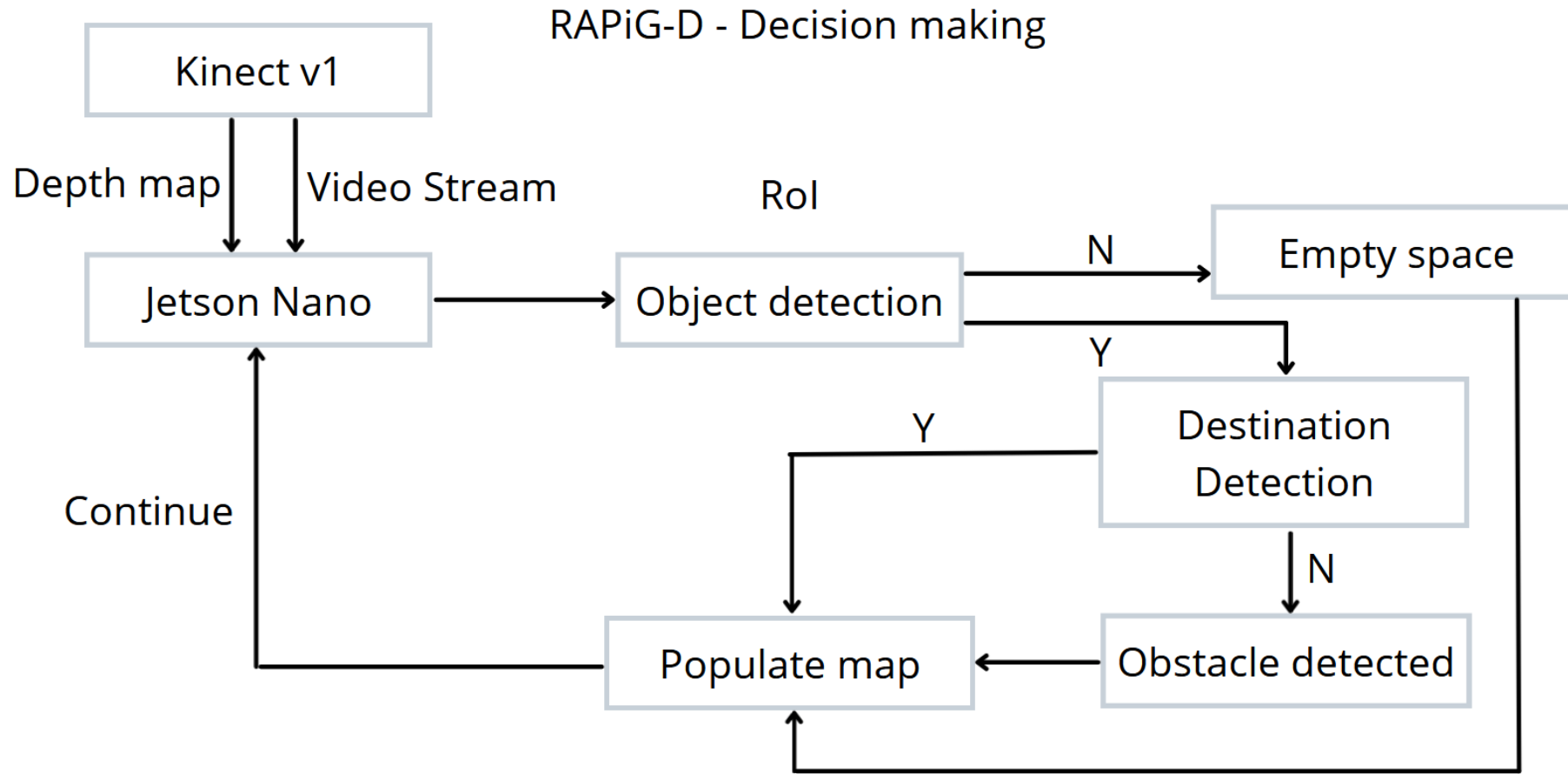
OBJECTIVES

- To perform hardware implementation of reinforcement learning based path planning to generate a map of an unknown environment by employing depth estimation based obstacle detection and analyze its performance.
- To estimate the best possible path to reach a given destination.

METHODOLOGY

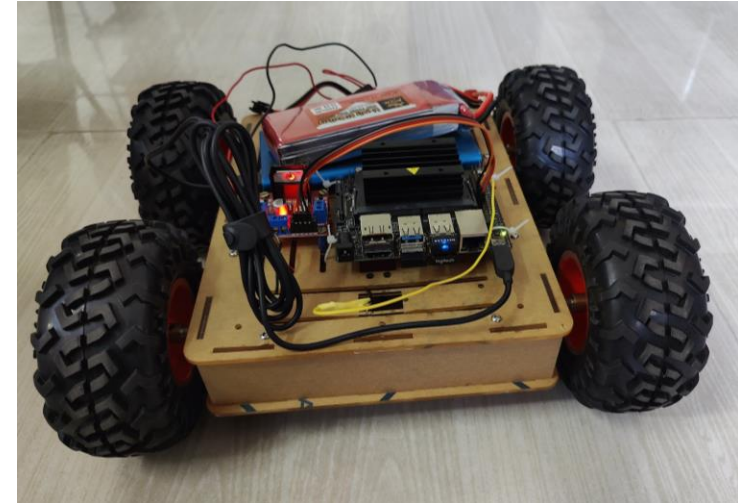


METHODOLOGY

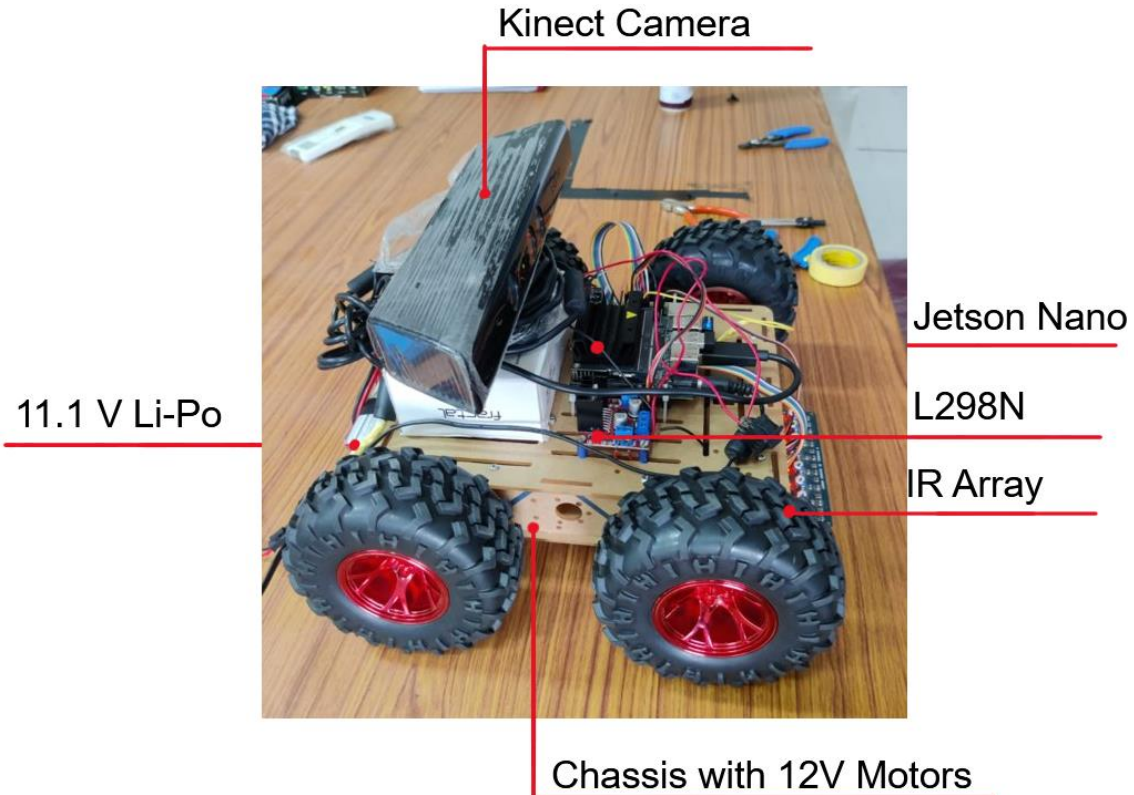


PREVIOUS PROGRESS

- Tried options for better weight distribution.
- Switched to Li-po battery instead of lead acid 12v battery.
- Set up web output for obstacle detection for remote inference
- Mounted and used IR sensor array to enable line following for movement calibration.
- Tried to calibrate turn using line following, timing and detecting line crossing
- Performed mapping of linear space with accurate obstacle detection



CURRENT BOT



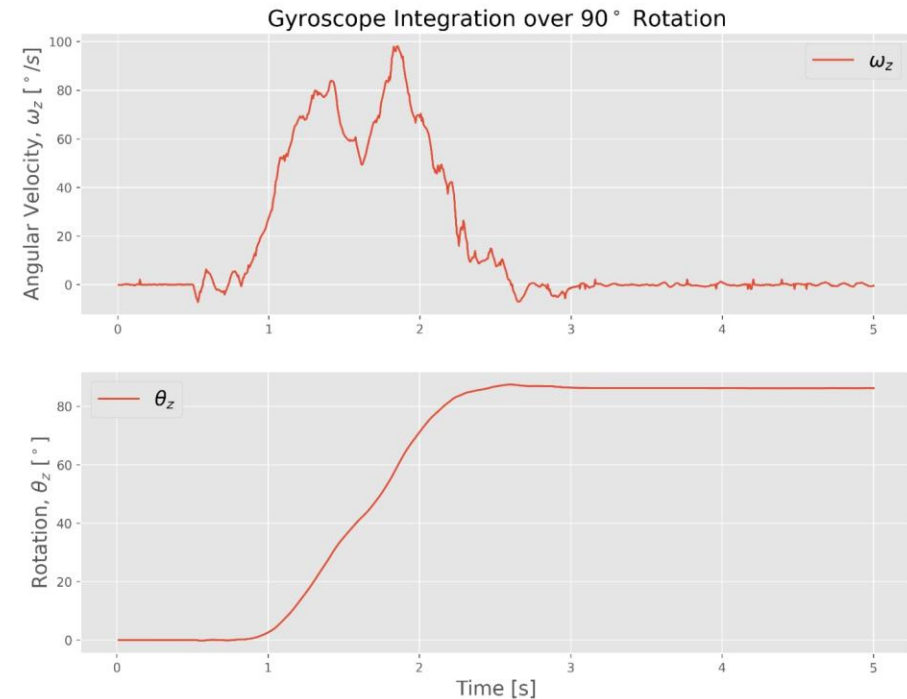
MPU6050
sensor

COMPONENTS USED

COMPONENTS	SPECIFICATION	NUMBER USED
Jetson Nano	Developer Kit - 4GB	1
Robot Chassis	Dimensions: 250 x 200 x 46 mm	1
DC motor	200 rpm 12V	2
Wheels	Diameter-130mm Width-60mm	4
Kinect v1 Camera	Stereo camera, IR camera	1
Motor Driver IC	L298N	1
MPU-6050	6 Axis motion tracking	1
Battery	11.1 V 4200mah Li-Po	1

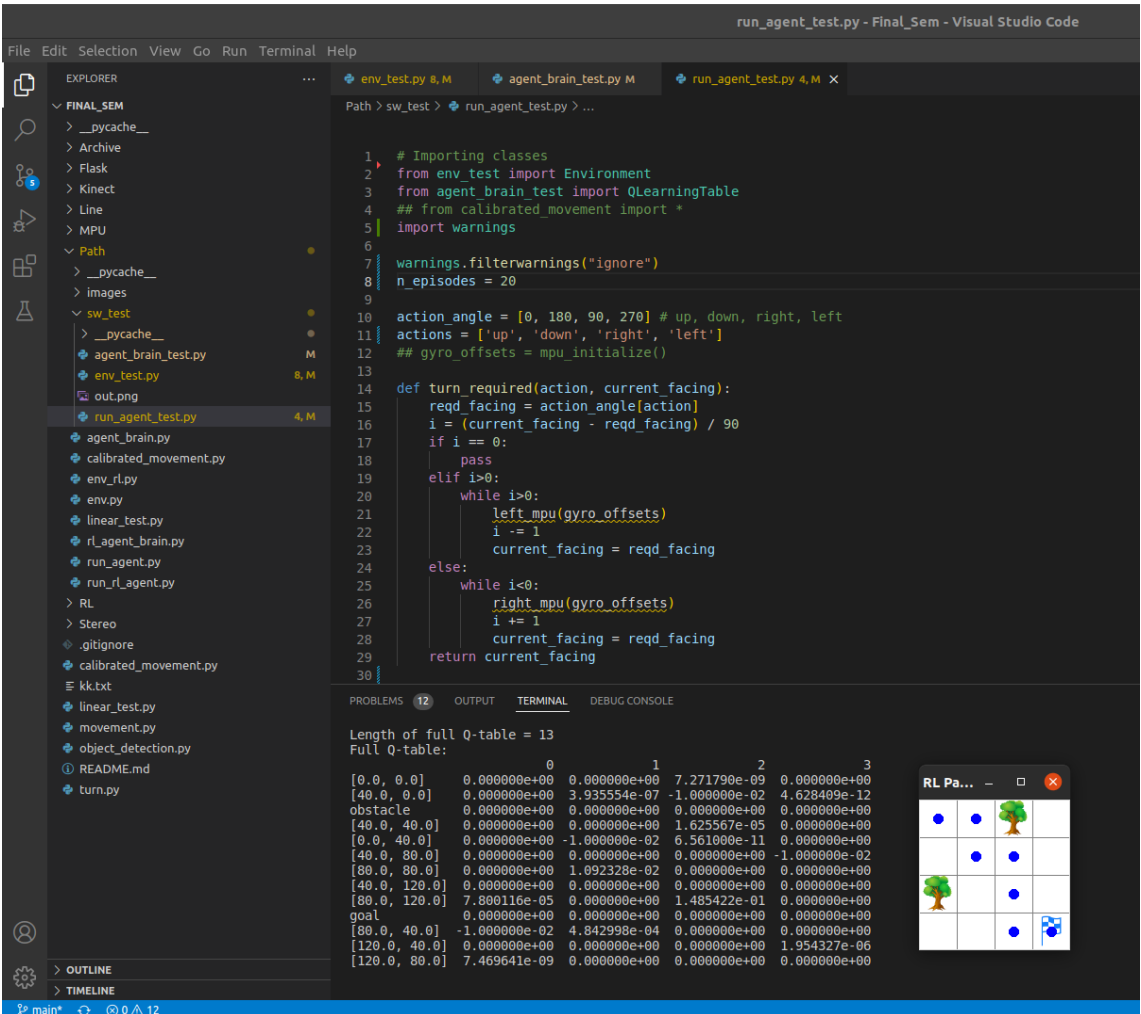
CURRENT PROGRESS

- Mounted stage for Kinect camera.
- Connected 100RPM 12V motors on all four wheels for better stability.
- Interfaced MPU-6050 with jetson nano.
- Integrated gyro output (Angular velocity) to get Angular displacement along z-axis which is suitable to identify bot turning.
- Used angular displacement destination to accurately turn left and right.



CURRENT PROGRESS

- Modified RL SARSA algorithm to make it suitable for hardware implementation by integrating the different modules of obstacle detection, calibrated movement and RL algorithm for path planning.
- Tested software simulation of the modified algorithm
- Tested the algorithm on a stage atop table with temporary obstacles.

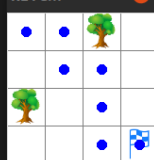


```
1 # Importing classes
2 from env_test import Environment
3 from agent_brain_test import QLearningTable
4 ## from calibrated_movement import +
5 import warnings
6
7 warnings.filterwarnings("ignore")
8 n_episodes = 20
9
10 action_angle = [0, 180, 90, 270] # up, down, right, left
11 actions = ['up', 'down', 'right', 'left']
12 ## gyro_offsets = mpu_initialize()
13
14 def turn_required(action, current_facing):
15     reqd_facing = action_angle[action]
16     i = (current_facing - reqd_facing) / 90
17     if i == 0:
18         pass
19     elif i > 0:
20         while i > 0:
21             left_mpu(gyro_offsets)
22             i -= 1
23             current_facing = reqd_facing
24     else:
25         while i < 0:
26             right_mpu(gyro_offsets)
27             i += 1
28             current_facing = reqd_facing
29     return current_facing
30
```

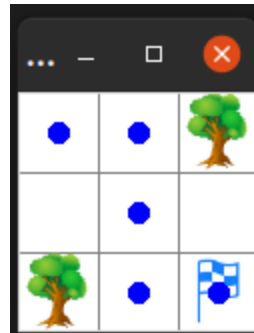
Length of full Q-table = 13
Full Q-table:

	0	1	2	3
[0.0, 0.0]	0.000000e+00	0.000000e+00	7.271790e-09	0.000000e+00
[40.0, 0.0]	0.000000e+00	3.935554e-07	-1.000000e-02	4.628409e-12
obstacle	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
[40.0, 40.0]	0.000000e+00	0.000000e+00	1.625567e-05	0.000000e+00
[0.0, 40.0]	0.000000e+00	-1.000000e-02	6.561000e-11	0.000000e+00
[40.0, 80.0]	0.000000e+00	0.000000e+00	0.000000e+00	-1.000000e-02
[80.0, 80.0]	0.000000e+00	1.092328e-02	0.000000e+00	0.000000e+00
[40.0, 120.0]	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
[80.0, 120.0]	7.800116e-05	0.000000e+00	1.485422e-01	0.000000e+00
goal	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
[80.0, 40.0]	-1.000000e-02	4.842998e-04	0.000000e+00	0.000000e+00
[120.0, 40.0]	0.000000e+00	0.000000e+00	0.000000e+00	1.954327e-06
[120.0, 80.0]	7.469641e-09	0.000000e+00	0.000000e+00	0.000000e+00

RL Pa... - [X]



CURRENT PROGRESS



Full Q-table:

	0	1	2	3
[0.0, 0.0]	0.00	0.000000	2.770883e-07	6.561000e-11
[40.0, 0.0]	0.00	0.000028	-1.000000e-02	3.254256e-10
[40.0, 40.0]	0.00	0.001828	0.000000e+00	0.000000e+00
[80.0, 40.0]	-0.01	0.000000	0.000000e+00	0.000000e+00
obstacle	0.00	0.000000	0.000000e+00	0.000000e+00
[0.0, 40.0]	0.00	-0.010000	0.000000e+00	0.000000e+00
[40.0, 80.0]	0.00	0.000000	6.793465e-02	0.000000e+00
goal	0.00	0.000000	0.000000e+00	0.000000e+00

Output of hardware implementation when tested atop table with temporary obstacles.

PROBLEMS FACED

- Current RL SARSA algorithm was not suitable for hardware implementation
- Compatibility issues when integrating the obstacle detection, calibrated movement and RL algorithm for path planning.
- Faced errors with processing MPU6050 raw data
- Issues with detecting and populating obstacles in map in hardware implementation

WORK PLAN

Third review:

- Test the validity of hardware implementation in proper testing space.
- Setup interface for mapping the environment remotely.
- Testing and optimizing.

QUESTIONS FROM PREVIOUS REVIEW:

1) Briefly explain your object detection methodology.

Ans: The bot detects objects using stereo vision and depth mapping. The bot detects objects at a distance of 45cm based on the threshold we have set. In order to check whether an object is an actual obstacle, we check if the pixels occupied by an object takes up more than 15% of the area of the pre-defined ROI.

2) Will the bot detect any object in its path?

Ans: Yes, the bot will detect any object in its path as an obstacle. Obstacle detection using depth mapping is very robust as it can detect objects of any shape and size that are at a particular distance from the camera.

3) Explain how you are powering the bot.

Ans: As of now, the four motors are powered by the Li-Poly battery. The Jetson NANO and the Kinect v1 are connected to an AC power source through adapters

QUESTIONS FROM PREVIOUS REVIEW:

4) What are the specifications of the motor you are using?

Ans: We are using four motors each rated for 12V and 100rpm.

5) Can the bot be connected to a wifi network?

Ans: Yes, the bot is capable of connecting to a network. We already pass commands to run the bot remotely using SSH.

CONFERENCE SUBMITTED

AIKP'22: Second International Conference On Artificial Intelligence and Knowledge Processing
Woxsen University
Hyderabad, India, July 22-23, 2022

REFERENCES

- Martin Gromniak and Jonas Stenzel , "Deep Reinforcement Learning for Mobile Robot Navigation", Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), July 2019
- Guillaume Sartoretti , Justin Kerr , Yunfei Shi, Glenn Wagner, T. K. Satish Kumar, Sven Koenig, and Howie Choset, "PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning", IEEE robotics and automation letters, , Vol. 4, no. 3, July 2019.
- Hyansu Bae , Gidong Kim , Jonguk Kim , Dianwei Qian 4 and Sukgyu Lee , "Multi-Robot Path Planning Method Using Reinforcement Learning", Multidisciplinary Digital Publishing Institute, Vol 3, No 4, May 2019.
- Jing Xin, Huan Zhao, Ding Liu, "Application of Deep Reinforcement Learning in Mobile Robot Path Planning", IEEE robotics and automation letters, Vol 2, No 3 , January 2019.

**END OF
PRESENTATION**

REINFORCEMENT LEARNING BASED AUTOMATED PATH PLANNING IN GARDEN ENVIRONMENT USING DEPTH - 'RAPIG-D'

THIRD REVIEW

- S SATHIYA MURTHI - 2018504604
- PRANAV BALAKRISHNAN - 2018504581
- C ROSHAN ABRAHAM - 2018504591

**BE ELECTRONICS AND COMMUNICATION
ENGINEERING**

**GUIDED BY:
DR. V. SATHIESH KUMAR,
ASSISTANT PROFESSOR,
DEPARTMENT OF ELECTRONICS
ENGINEERING
MIT CAMPUS, ANNA UNIVERSITY**

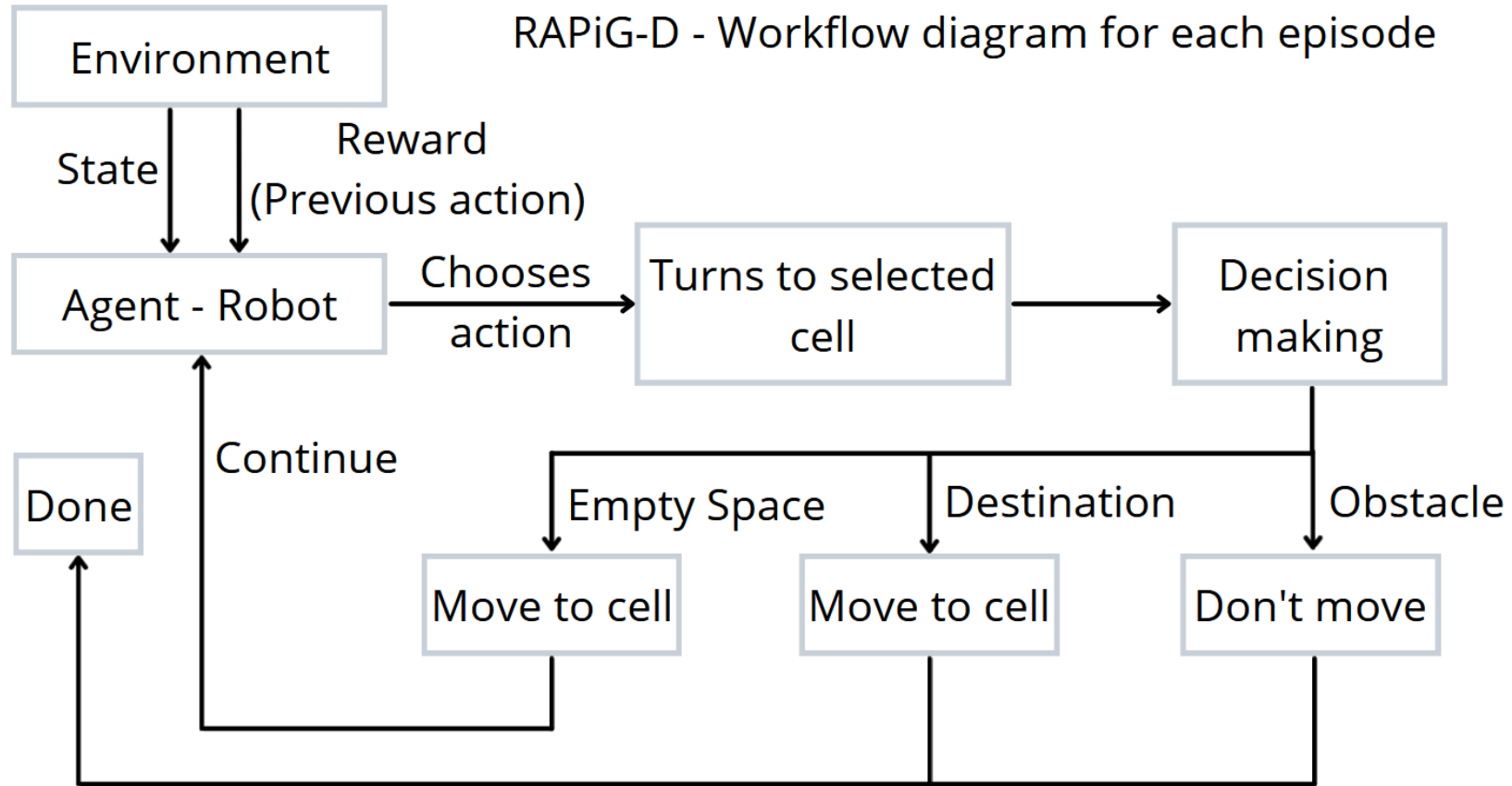
TABLE OF CONTENTS

- Objectives
- Methodology / Block Diagram
- Previous progress Recap
- Current robot design
- Current Progress
- Code / Demo
- References

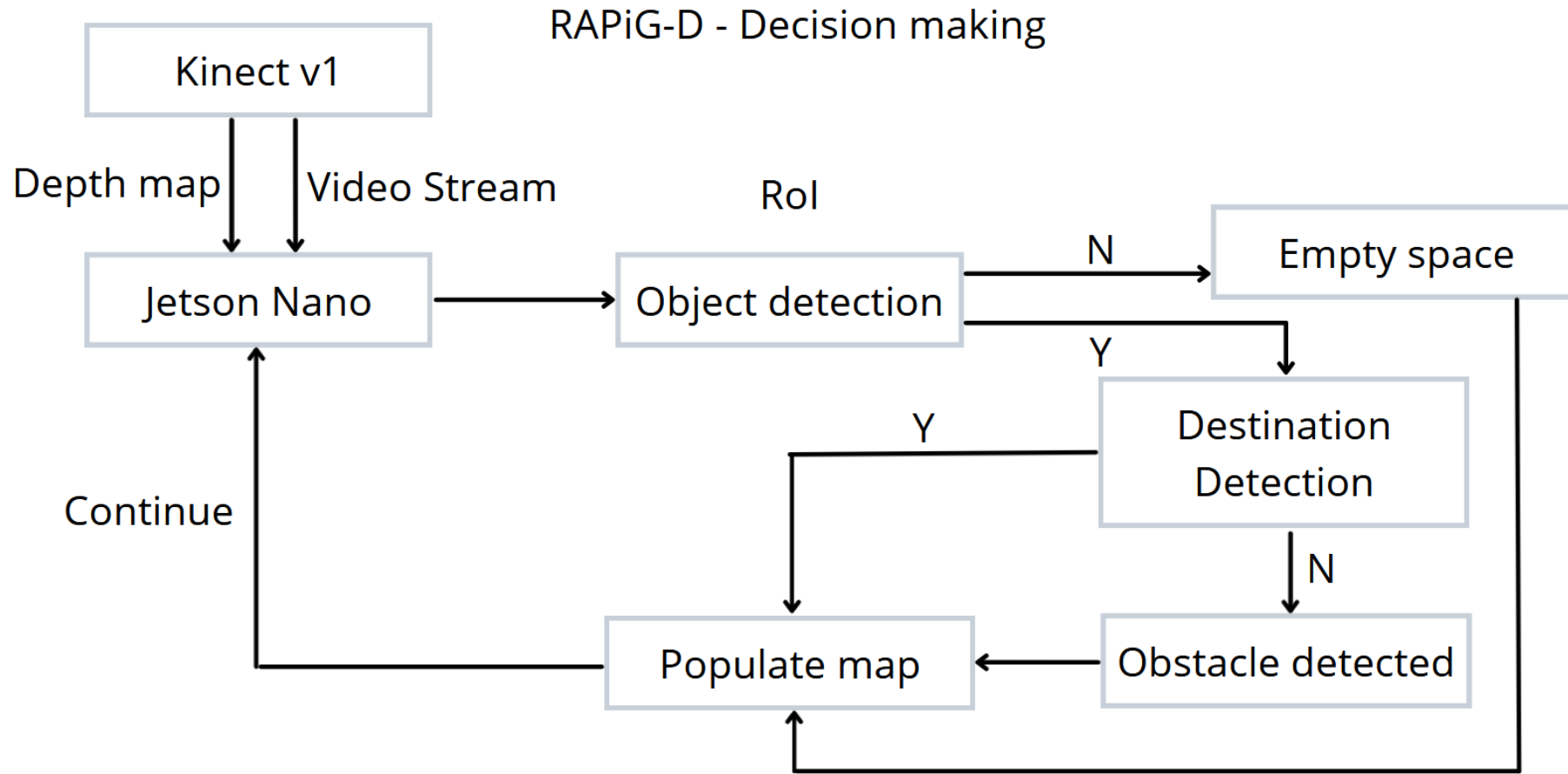
OBJECTIVES

- To perform hardware implementation of reinforcement learning based path planning to generate a map of an unknown environment by employing depth estimation based obstacle detection and analyze its performance.
- To estimate the best possible path to reach a given destination.

METHODOLOGY



METHODOLOGY

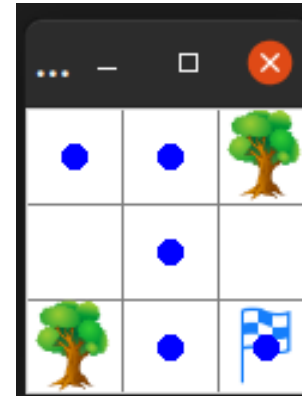


PREVIOUS PROGRESS

- Calibrated bot movement using MPU 6050.
- Modified RL SARSA algorithm to make it suitable for hardware implementation.
- Tested software simulation of the modified algorithm
- Tested the algorithm on a stage atop table with temporary obstacles.

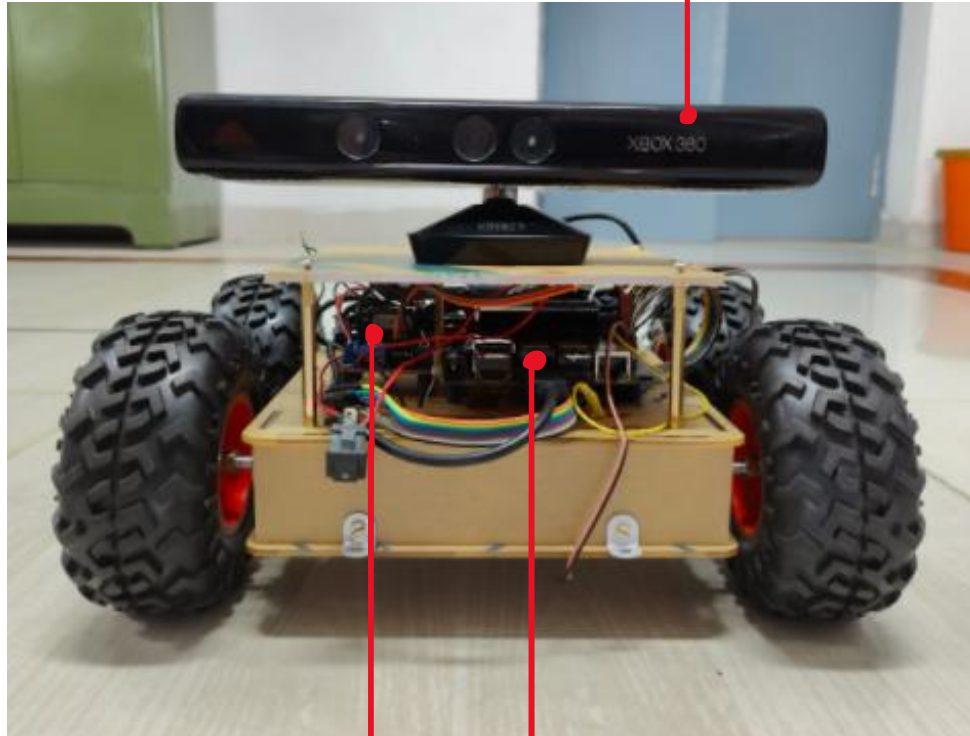
Full Q-table:

	0	1	2	3
[0.0, 0.0]	0.00	0.000000	2.770883e-07	6.561000e-11
[40.0, 0.0]	0.00	0.000028	-1.000000e-02	3.254256e-10
[40.0, 40.0]	0.00	0.001828	0.000000e+00	0.000000e+00
[80.0, 40.0]	-0.01	0.000000	0.000000e+00	0.000000e+00
obstacle	0.00	0.000000	0.000000e+00	0.000000e+00
[0.0, 40.0]	0.00	-0.010000	0.000000e+00	0.000000e+00
[40.0, 80.0]	0.00	0.000000	6.793465e-02	0.000000e+00
goal	0.00	0.000000	0.000000e+00	0.000000e+00



Output of hardware implementation when tested atop table with temporary obstacles.

CURRENT BOT



Kinect v1 Camera



MPU6050 sensor

NVIDIA Jetson Nano

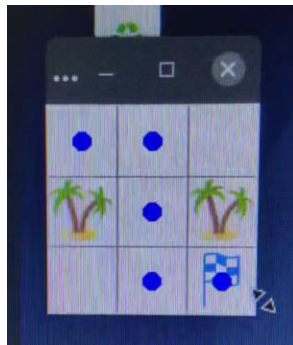
L298N Driver

COMPONENTS USED

COMPONENTS	SPECIFICATION	NUMBER USED
Jetson Nano	Developer Kit - 4GB	1
Robot Chassis	Dimensions: 250 x 200 x 46 mm	1
DC motor	200 rpm 12V	2
Wheels	Diameter-130mm Width-60mm	4
Kinect v1 Camera	Stereo camera, IR camera	1
Motor Driver IC	L298N	1
MPU-6050	6 Axis motion tracking	1
Battery	11.1 V 4200mah Li-Po	1

CURRENT PROGRESS

- Optimised bot movement.
- Performed hardware test in testing space
- Set up web app for passing information regarding action chosen in each step.
- Performed 2D space mapping in testing space successfully.
- Ran 10 episodes on 3x3 space and determined the optimal path.



```
jetson@nano: ~$  
The shortest route: 4  
The longest route: 10  
[40.0, 0.0]  
[40.0, 40.0]  
[40.0, 80.0]  
[80.0, 80.0]
```



```
Length of final Q-table = 3  
Final Q-table with values from the final route:  
          0          1          2          3  
[40.0, 0.0]  0.0  8.100000e-07  0.0000  0.0000  
[40.0, 40.0]  0.0  9.000000e-05 -0.0199 -0.0199  
[40.0, 80.0]  0.0  0.000000e+00  0.0199  0.0000  
  
Length of full Q-table = 7  
Full Q-table:  
          0          1          2          3  
[0.0, 0.0]   0.0 -1.990000e-02  0.0000  0.0000  
[40.0, 0.0]  0.0  8.100000e-07  0.0000  0.0000  
[80.0, 0.0]  0.0 -1.990000e-02  0.0000  0.0000  
[40.0, 40.0]  0.0  9.000000e-05 -0.0199 -0.0199  
obstacle    0.0  0.000000e+00  0.0000  0.0000  
[40.0, 80.0]  0.0  0.000000e+00  0.0199  0.0000  
goal        0.0  0.000000e+00  0.0000  0.0000
```

CODE / DEMO – OBSTACLE DETECTION

```
45 def show_depth(): ## 640x480
46     global threshold
47     global current_depth
48
49     depth, timestamp = freenect.sync_get_depth()
50     depth = 255 * np.logical_and(depth >= 0,
51                                 depth <= current_depth + threshold)
52     depth = depth.astype(np.uint8)
53     depth = cv2.cvtColor(depth, cv2.COLOR_GRAY2RGB)
54
55     is_obj = check_if_object(depth)
56     if is_obj:
57         color = (0, 0, 255)
58         print('Object')
59         try:
60             send_to_flask(True)
61         except Exception as e:
62             pass
63         time.sleep(1)
64     else:
65         color = (0, 255, 0)
66         try:
67             send_to_flask(False)
68         except Exception as e:
69             pass
70
71     frame = cv2.rectangle(depth, ROI[0], ROI[1], color, 1)
72
73
74     cv2.imshow('Depth', frame)
75
```

```
40 def check_if_object(frame):
41     area = int((1 - 2*roi_val) * height * (1 - 2*roi_val) * width)
42     pixels = int(np.sum(frame[ROI[0][1]:ROI[1][1], ROI[0][0]:ROI[1][0], 0]) / 255)
43     return pixels > (area * alert_thresh)
```

Libraries used:

- Freenect
- OpenCV
- freenect.sync_get_depth() - Function used to generate depth map

CODE / DEMO – CALIBRATED MOVEMENT

```
58 def left_mpu(gyro_offsets):
59     stop()
60     angle = 0
61     rot_axis = 2 # axis being rotated (2 = z-axis)
62     data, t_vec = [], []
63     t0 = time.time()
64     while angle < angle_thresh:
65         left()
66         data.append(get_gyro())
67         t_vec.append(time.time()-t0)
68         data_offseted = np.array(data)[: , rot_axis]-gyro_offsets[rot_axis]
69         integ1_array = cumtrapz(data_offseted, x=t_vec)
70         try:
71             angle = integ1_array[-1]
72         except:
73             pass
74     stop()
```

- Turn calibrated using MPU6050 gyro sensor data integration
- Linear movement controlled by timing
- Motor control using GPIO pins

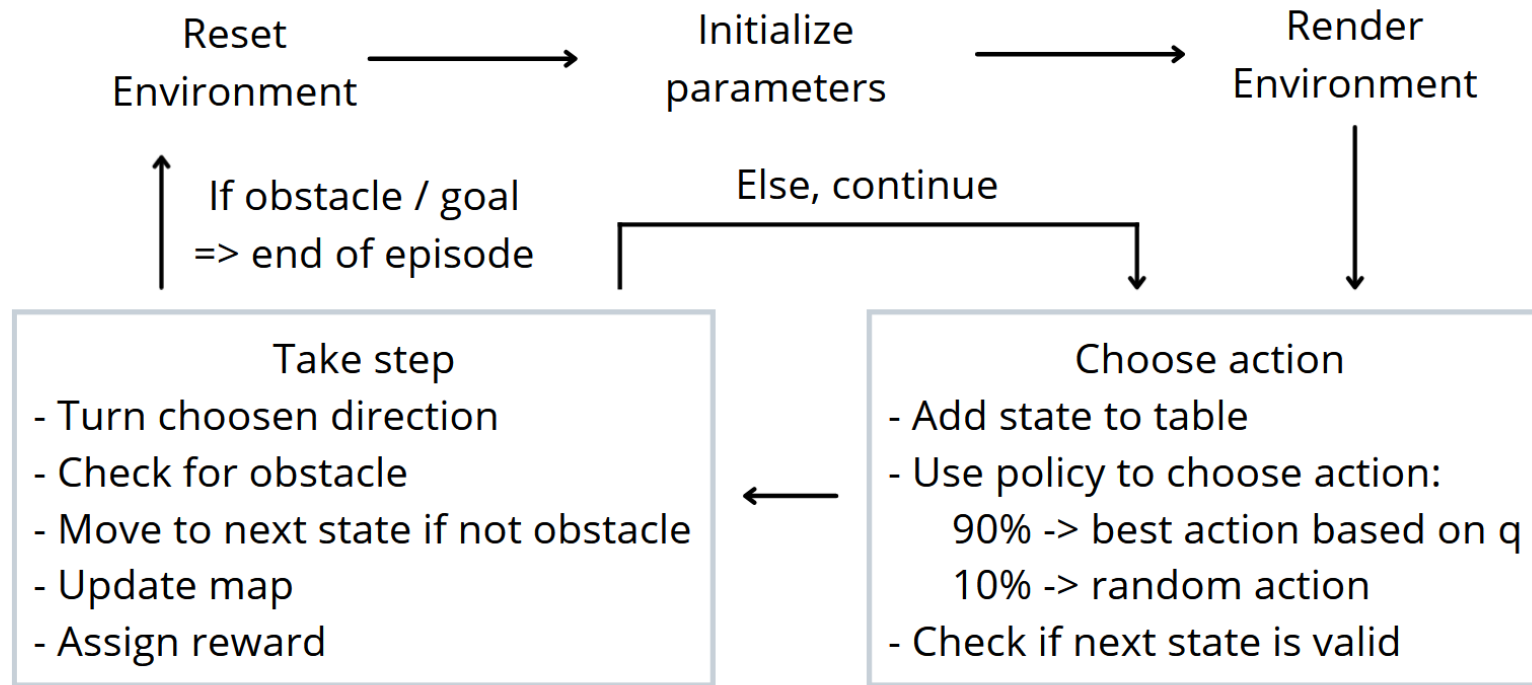
```
187 def move_one_f():
188     fwd()
189     time.sleep(1.32)
190     stop()
191     time.sleep(1)
192
193
194 def move_one_b():
195     back()
196     time.sleep(1.32)
197     stop()
198     time.sleep(1)
```

```
46 def fwd():
47     stop()
48     GPIO.output(motor_pin_d, GPIO.HIGH)
49     GPIO.output(motor_pin_c, GPIO.HIGH)
50     GPIO.output(motor_pin_a, GPIO.LOW)
51     GPIO.output(motor_pin_b, GPIO.LOW)
52
53
54 def back():
55     stop()
56     GPIO.output(motor_pin_a, GPIO.HIGH)
57     GPIO.output(motor_pin_b, GPIO.HIGH)
58     GPIO.output(motor_pin_c, GPIO.LOW)
59     GPIO.output(motor_pin_d, GPIO.LOW)
```

```
61 def left():
62     stop()
63     GPIO.output(motor_pin_c, GPIO.HIGH)
64     GPIO.output(motor_pin_b, GPIO.HIGH)
65     GPIO.output(motor_pin_a, GPIO.LOW)
66     GPIO.output(motor_pin_d, GPIO.LOW)
67
68 def right():
69     stop()
70     GPIO.output(motor_pin_d, GPIO.HIGH)
71     GPIO.output(motor_pin_a, GPIO.HIGH)
72     GPIO.output(motor_pin_b, GPIO.LOW)
73     GPIO.output(motor_pin_c, GPIO.LOW)
```

CODE / DEMO – RL PATH PLANNING

RL Path Planning - each episode



CONFERENCE ACCEPTANCE

AIKP'22
Second International Conference On
Artificial Intelligence and Knowledge Processing
Woxsen University
Hyderabad, India, July 22-23, 2022

AKIP'22 notification for paper 8079  [Inbox x](#)



AKIP'22 <akip22@easychair.org>
to me ▾

Kindly submit the revised paper with the registration fee on or before 20th June 2022 in easy-chair itself.

SUBMISSION: 8079

TITLE: Reinforcement Learning based Automated Path Planning in Garden environment using Depth - RAPIG-D

----- REVIEW 1 -----

SUBMISSION: 8079

TITLE: Reinforcement Learning based Automated Path Planning in Garden environment using Depth - RAPIG-D

AUTHORS: Sathiya Murthi S, Pranav Balakrishnan, Roshan Abraham C and Sathiesh Kumar V

----- Overall evaluation -----

SCORE: 2 (accept)

----- TEXT:

Dear Authors,

Congratulations!

Your paper has been accepted to the AKIP'22 Conference.

REFERENCES

- Martin Gromniak and Jonas Stenzel , "Deep Reinforcement Learning for Mobile Robot Navigation", Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), July 2019
- Guillaume Sartoretti , Justin Kerr , Yunfei Shi, Glenn Wagner, T. K. Satish Kumar, Sven Koenig, and Howie Choset, "PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning", IEEE Robotics and Automation Letters, Vol. 4, no. 3, July 2019.
- Hyansu Bae , Gidong Kim , Jonguk Kim , Dianwei Qian and Sukgyu Lee , "Multi-Robot Path Planning Method Using Reinforcement Learning", Multidisciplinary Digital Publishing Institute, Vol 3, No 4, May 2019.
- Jing Xin, Huan Zhao, Ding Liu, "Application of Deep Reinforcement Learning in Mobile Robot Path Planning", IEEE Robotics and Automation Letters, Vol 2, No 3 , January 2019.

**END OF
PRESENTATION**