MULTIPLE ROBOTS PATH PLANNING BASED ON REINFORCEMENT LEARNING FOR OBJECT TRANSPORTATION

by

Sarawat Ruamngern

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Mechatronic

Examination Committee:	Prof. Manukid Parnichkun (Chairperson)
	Prof. Matthew N. Dailey
	Dr. Mongkol Ekpanyapon

Nationality:	Thai
Previous Degree:	Bachelor of Engineering in Mechanical
	Chulalongkorn University
	Bangkok, Thailand
Scholarship Donor:	His Majesty the King's Scholarships (Thailand)

Asian Institute of Technology
School of Engineering and Technology
Thailand
May 2022

AUTHOR'S DECLARATION

I, Sarawat Ruamngern, declare that the research work carried out for this thesis was in accordance with the regulations of the Asian Institute of Technology. The work presented in it are my own and has been generated by me as the result of my own original research, and if external sources were used, such sources have been cited. It is original and has not been submitted to any other institution to obtain another degree or qualification. This is a true copy of the thesis, including final revisions.

Date: 5 May 2022

Name (in printed letters): Sarawat Ruamngern

Signature:

ACKNOWLEDGMENTS

First and foremost, I would like to express my thankfulness to my advisor, Prof. Manukid Parnichkun, for his invaluable assistance such as ideas, suggestions, and budget supportation during the thesis process. It would be difficult to complete this thesis without his guidance.

Secondly, I would like to express my appreciation to the members of the examination committee, Prof. Matthew N. Dailey and Assoc.Prof. Mongkol Ekpanyapon, for their recommendations.

In addition, I would like to extend my thankfulness to Ms. Alisa Kunapinun, the AIT senior student in Mechatronics field. She generously provided me the ideas, suggestions, knowledges, and computing machine. To achieve this thesis, the high-capacity computing machine is required, and my machine could not reach that specification for model training process. Therefore, all the training processes were accomplished by her machine.

I also would like to show my gratitude to His Majesty the King's Scholarship of Thailand for funding my whole master's degree and providing me the opportunity to have a great experience in AIT.

Finally, I would like to thank my parents for their supportation and aid during my education which make me could fully concentrate on my study.

ABSTRACT

This thesis proposed reinforcement learning method to perform the object transportation task for multiple robots. This task consists of two main processes, path planning and motion control task. Double deep Q-learning (DDQN) model is selected to achieve path planning for any random environment. To increase the capability of reinforcement learning model, semi-supervised method by A* algorithm is applied during the training process in order to find the optimal path. For motion control task, reinforcement learning model must control a motion of differential wheeled mobile robot by providing actions composed of speeds of left wheel and right wheel. The models are separately trained for two different purposes. The first agent is trained to deal with path following task and another agent is trained to handle with point following task. The agent of point following task is used to control the group of robots to move with a fixed group shape. The agent which is trained in the environment without disturbance is used in simulation. And the agent which is trained in the environment included disturbance is applied in practice with three differential wheeled mobile robots. Proximal policy optimization (PPO) is selected to achieve path following task in simulation and point following task in practice. Deep deterministic policy gradient (DDPG) is selected to complete path following task in practice. And soft actor-critic is selected to complete point following task in simulation. Finally, the integration of proposed reinforcement learning models can accomplish the object transportation task for multi-robot system appropriately both in simulation and practice.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figures **Page**

LIST OF ABBREVIATIONS

A*	= Astar
AI	= Artificial Intelligent
AIT	= Asian Institute of Technology
CNN	= Convolution Neural Network
CPU	= Central Processing Unit
DDPG	= Deep Deterministic Policy Gradient
DDQN	= Double Deep Q-learning Network
DQN	= Deep Q-learning Network
GA	= Genetic Algorithm
GPU	= Graphic Processing Unit
MDP	= Markov decision process
MIR	= Mobile Industrial Robot
ML	= Machine Learning
MRs	= Multi-Robot system
NN	= Neural Network
OS	= Operation System
RAM	= Random Access Memory
RL	= Reinforcement Learning
PID	= Proportional-Integral-Derivative
PPO	= Proximal Policy Optimization
SAC	= Soft Actor-Critic

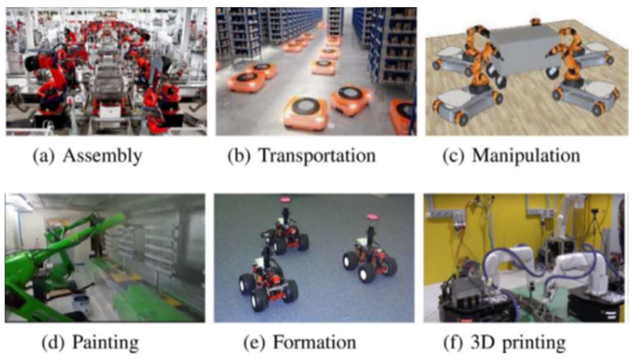
CHAPTER 1
INTRODUCTION

1.1 Background of the Study

In many cases, it is obviously that multi-robot cooperation can perform complex tasks more effectively than one single powerful robot for tasks that impossible for one robot to accomplish. In industrial application, assembling task, transporting task, painting task, packaging task, and welding task usually require flexibility to perform (see Figure 1.1). Hence, the cooperation of multi-robot system has been developed with many techniques for increasing efficiency and enhancing flexibility in industrial application.

Figure 1.1

Application Scenarios of Collaborative Robotic Manipulation (Feng et al., 2020)

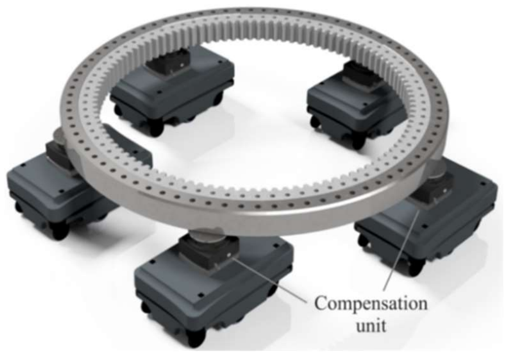


The fundamental of multi-robot system is to distribute subtasks to individual robots and allow them to handle the tasks to solve the complex problem. The multi-robot system (MRs) is very cost effective when compared to one single powerful robot. The comfortable of multi-robot system makes them popular widely in various application. In contrast, multi-robot system requires a lot of setting or tunning for deployment and has high possibility to collision to each other as well.

This thesis focuses on path planning with formation forming of multiple mobile robots for object transportation. Formation forming of group of mobile robots is often used to extend the capability of a single mobile robot to break down and distribute the complex task into smaller subtasks (see Figure 1.2). The connectivity of these mobile robots enables them to carry out the task simpler than one capability robot. Moreover, formation forming increase the flexibility of overall system to adapt with complexed and customized of large objects.

Figure 1.2

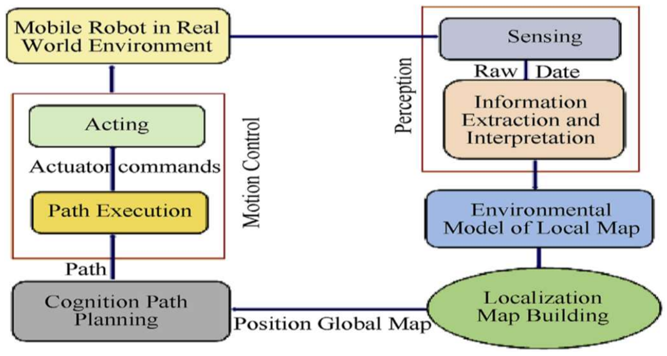
Mobile Industrial Robots Carrying a Crane Bearing (Recker et al., 2020)



For object transportation by multi-robot system, one important problem is path planning for group of mobile robots. There is various research proposed efficient path planning algorithm to establish path and movement that allow mobile robots to reach the goal autonomously and safety in working environment with collision-free (Patle et al., 2019) (see Figure 1.3). Most approach try to generate optimal path for navigating a mobile robot to move from initial location to final location by avoiding static and dynamic obstacles presented in the environment.

Figure 1.3

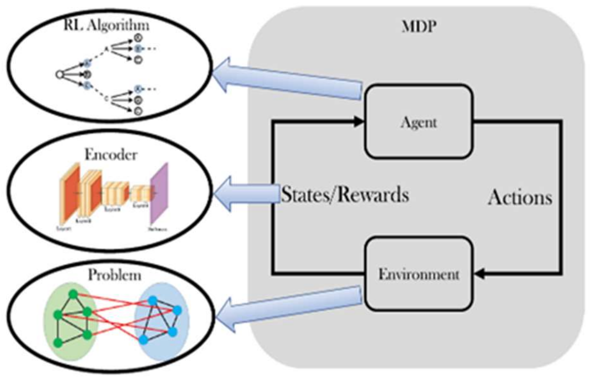
Flow Diagram for Mobile Robot Navigation (Patle et al., 2019)


Mobile Robot in Real World Environment
Sensing
Raw Date
Perception
Information Extraction and Interpretation
Acting
Actuator commands
Path Execution
Motion Control
Environmental Model of Local Map
Path
Cognition Path Planning
Position Global Map
Localization Map Building

At present, there are several studies proposed machine learning approach in order to solve path planning problem. Reinforcement learning (RL) algorithm is one of the most popular methods in machine learning approach which be proposed to solve path planning problem for mobile robot navigation (Sartoretti et al., 2019) (Wang et al., 2020). Reinforcement learning method proposes an automatically search of agent for optimization problem by training an agent in self-supervised. To apply reinforcement learning method, we design the model of problem as a sequential of decision-making process. Then the agent can interact to the environment by performing actions sequentially to find the optimal solution (see Figure 1.4). Markov decision process (MDP) provides a mathematical framework for the model in this type of problem (Mazyavkina et al., 2021).

Figure 1.4

Concept of Reinforcement Learning (Mazyavkina et al., 2021)



1.2 Statement of the Problem

There are various algorithms have been proposed for solving path planning problem. Recently, one of interesting emerging trend to solve this optimization problem is machine learning approach. To increase automatically learning capability and making-decision of robots, reinforcement learning method is one branch of machine learning that allow agent to learn autonomously how to accomplish any task even the environment has changed with less setting or tunning parameters when compared to traditional methods.

When considering the object transportation for multi-robot system, it is interesting to explore how reinforcement learning approach handles to this problem. Therefore, this thesis proposed reinforcement learning models to perform path planning and motion control task for multi-robot system. In this thesis, the agent which is trained to solve path planning problem must find the optimal path with obstacles avoidance for individual robots from initial positions to loading positions. It also find the optimal path for the group of robots from loading point to the delivery point. Another agent which is trained for motion control task must control the movement of multiple robots to follow their paths appropriately. The integration of these agents is used to perform the object transportation task. The performance of reinforcement learning method is evaluated both in simulation and practice.

1.3 Objectives of the Study

The main objective of this thesis is to develop reinforcement learning models to solve the object transportation problem for multi-robot system. To achieve the objective, the following sub-objectives are required.

1. Develop path planning model based on reinforcement learning method.
2. Develop motion control model for differential wheeled mobile robot based on reinforcement learning method.
3. Develop three differential wheeled mobile robots to implement the models in practice.

1.4 Scope and Limitation

The scopes of this thesis are listed as following:

1. Reinforcement learning method is used for path planning.
2. Reinforcement learning method is used for motion control task.
3. Three differential-wheeled mobile robots are developed to evaluate the performance of reinforcement learning models in practice.
4. Translation and rotation of mobile robot limited in cartesian plane (X, Y, θ).
5. The working area is enveloped within 1.2 x 2.0 square meter.
6. The performance of reinforcement learning models will be compared among value-based method and policy-based method.

CHAPTER 2
LITERATURE REVIEW

There are several published papers involved with the multi-robot cooperation. As mentioned in the previous chapter, this thesis focuses on optimal path planning with formation forming for object transportation. Therefore, in this chapter, the related literatures will be reviewed separated into following sections such that 1) multi-robot system 2) localization of indoor mobile robot, 3) path planning for mobile robot, 4) formation control for nonholonomic robot, 5) reinforcement learning with optimization problem.

2.1 Multi-Robot System

There is a review of research in multi-robot system. Goh & Tjahjono (2006) mentioned that multi-robot system has potential to solve vast application of complex tasks in different domain. Much research has been proposed coordination algorithm inspired from small living organisms such as birds, fishes, or insects. The algorithm can divide into leader-follower or leaderless approaches. Certain communication is appropriate in a situation that real time response is not necessary such as path planning, formation forming. The communication between multi-agents is suitable for the situation when it required immediate reaction such as obstacle detection, collision avoidance.

The architecture of multi-robot system is very importance according to applications. The architecture of control system can be separated into centralized or decentralized. In centralized architecture, there is only single agent provide command and supervise to the other agents to perform the cooperative tasks. This type of architecture has highly efficient when control a small group of robots (Khoshnevis & Bekey, 1998). In centralized architecture,   However, this type of architecture is difficult to handle with large number of robots system. On the other hand, decentralized architectures do not have a leader agent that commands the overall system. This architecture applies behavior-based control over multiple robots instead. Decentralized architecture treats all agents equally and respect each agent to take control itself individually.

2.2 Localization of Indoor Mobile Robot

Localization is a technique which necessary for mobile robots to know their current position and use this data to generate their next movement. Most localization technique provide the robot, a measurement sensor that allow mobile robot getting environment information. However, those technique require high cost as they use much equipment. The mobile robot became expensive and complicate body structure. Several research of localization carry out this problem by using external sensors getting the environment information. Shim & Cho (2015) proposed alternative technique to localize the indoor mobile robot by using two surveillance cameras. They applied HSV range technique for the images from different perspective in order to remove the object's shadow. And using homography technique to create two-dimensional map with object information (see Figures 2.2, 2.3)

Figure 2.1

Two Images Viewed from Two Surveillance Cameras at Indoor (Shim & Cho, 2015)

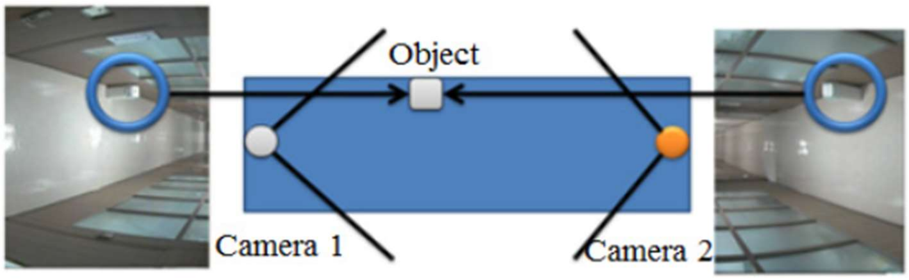
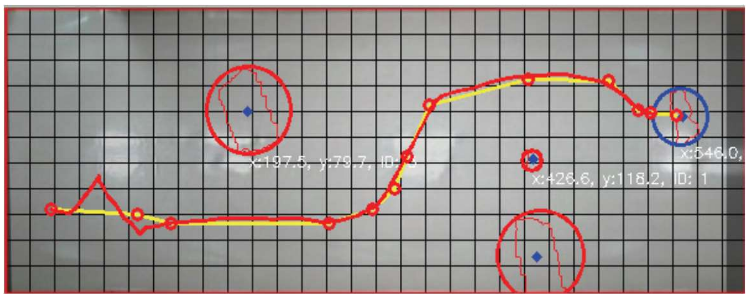


Figure 2.2

Robot's Path from the External Cameras Localization Method (Shim & Cho, 2015).

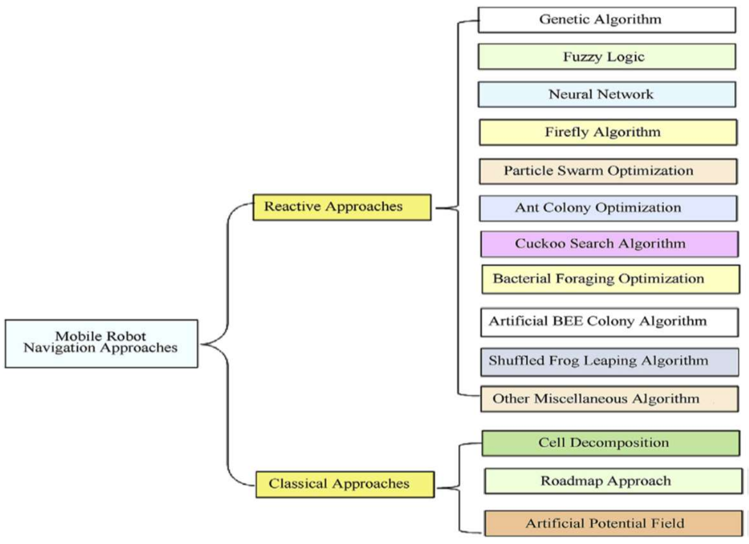


2.3 Path Planning for Mobile Robot

Currently, many researchers developed and proposed several techniques to perform a mobile robot navigation. The strategy of navigation can be classified to global navigation and local navigation based on required information of environment for path planning. (Patle et al., 2019). For global navigation, overall information of environment is required for the mobile robot such as current position, goal position, or obstacle position. In global navigation, many path planning algorithms based on classical approach are developed such as artificial potential field (APF), roadmap approach (RA), and cell decomposition (CD). In contrast, local navigation is concerned where the mobile robot handle with the unknow or partial known environment. The path planning algorithm for local navigation is based on reactive approach such as neural network (NN), fuzzy logic (FL), genetic algorithm (GA) etc., (see Figure 2.7).

Figure 2.3

Classification of Mobile Robot Navigational Approaches (Patle et al., 2019)


Mobile Robot Navigation Approaches
Reactive Approaches
Genetic Algorithm
Fuzzy Logic
Neural Network
Firefly Algorithm
Particle Swarm Optimization
Ant Colony Optimization
Cuckoo Search Algorithm
Bacterial Foraging Optimization
Artificial BEE Colony Algorithm
Shuffled Frog Leaping Algorithm
Other Miscellaneous Algorithm
Classical Approaches
Cell Decomposition
Roadmap Approach
Artificial Potential Field

One of the most well-known searching methods for path planning problem is A* algorithm. This algorithm can be applied on grid map or topological configuration space (Duchon et al., 2014). This algorithm finds the shortest path by applying heuristic searching. In A* algorithm, each cell of the configuration space is evaluated by the loss value:

$$f(v) = h(v) + g(v) \tag{2.1}$$

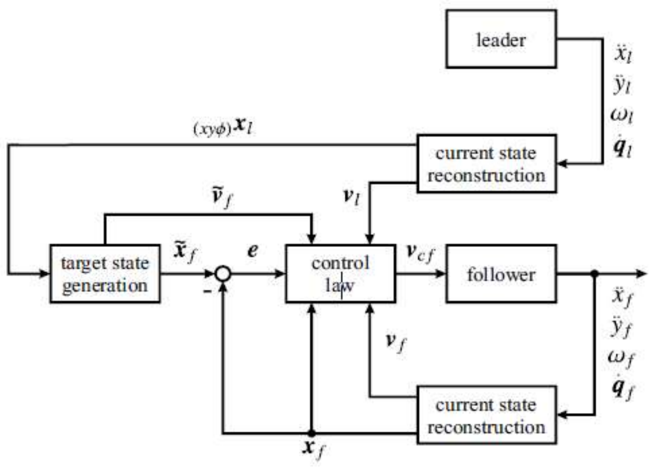
Where $f(v)$ is the loss value that will be evaluated for each adjacent cell connected to the current cell, $h(v)$ is Euclidean distance or heuristic distance between the current cell and the goal state, and $g(v)$ is the length of the path from the initial state to the current state according to the selected sequence of cells. The cell which has the lowest loss value will be chosen as the next cell in the sequence. This algorithm continuously searches and evaluates the cell until it reaches the goal state in order to obtain the lowest loss value.

2.4 Formation Control for Nonholonomic Robot

Formation control is a one of application for multi-robot cooperation, use to extend capability of a single robot. Multi-robot formation control increases flexibility of the overall system and adaptability for object transportation. Multi-robot formation control can be separated into holonomic and nonholonomic robot. This thesis focuses on formation control for nonholonomic mobile robot which is more complex and more practical in industrial usability. Recker et al. (2020) presented a comparison and evaluation of leader-follower based formation control approach for mobile robots within transportation process (see Figure 2.4).

Figure 2.4

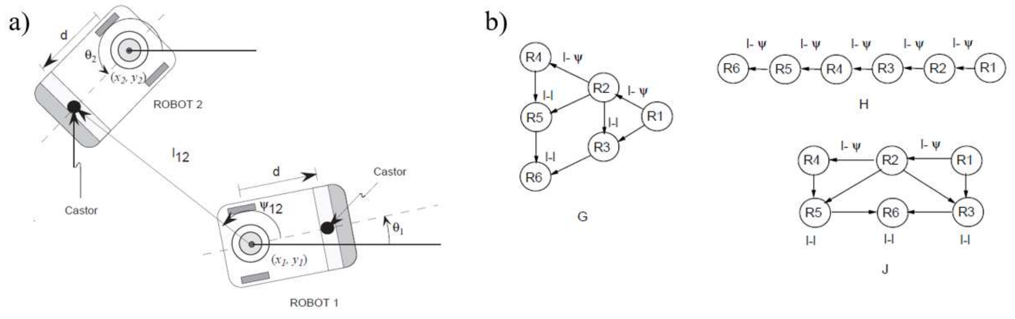
Control Structure of Leader-Follower (Recker et al., 2020)



Desai et al. (1998) presented the l- ψ -control law (l.c.). Within this control law, the mobile robot's state is considered both the distance and the angle between a leader and the follower. According to this state representation of the mobile robots, the linear velocity and angular velocity of a follower are obtained by the linearization method (see Figure 2.5).

Figure 2.5

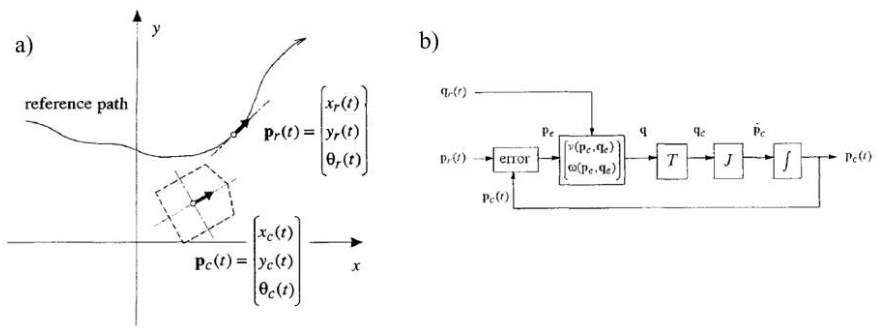
l- ψ-Control a) Notation, b) Isomorphic Diagraphs (Desai et al., 1998)



Kanayama, (1990) presented the cartesian control (c.c.) which the robot's state is represented in the cartesian state (x, y, θ) for all members in a formation. the linear control velocities and angular control velocities of a particular following robots are calculated by Lyapunov stabilizing the error dynamics of the mobile robot (see Figure 2.6).

Figure 2.6

Cartesian Control a) Postures, b) Tracking Controller (Kanayama, 1990)


a)
y
reference path
$\mathbf{p}_r(t) = \begin{bmatrix} x_r(t) \\ y_r(t) \\ \theta_r(t) \end{bmatrix}$
$\mathbf{p}_c(t) = \begin{bmatrix} x_c(t) \\ y_c(t) \\ \theta_c(t) \end{bmatrix}$
x
b)
$q_r(t)$
$p_r(t)$
error
p_e
$v(p_e, q_r)$
$\omega(p_e, q_r)$
q
T
q_c
J
$\dot{p}_c$
$\int$
$p_c(t)$
$p_c(t)$

2.5 Reinforcement Learning (RL) with Optimization Problem

Optimization problems are involved with finding optimal value among different possibility choices. In mobile robot path planning problem, finding the shortest path among all available paths is concerned. Reinforcement learning is one of the emerging trends to solve optimization problem by training a machine learning algorithm. Mazyavkina et al. (2021) presented a survey of recent applying of reinforcement learning framework to solve optimization problem. To apply reinforcement learning method, we design the model of problem as a sequential of decision-making process. Then the agent can interact to the environment by performing actions sequentially to find the optimal solution. Markov decision process (MDP) is the important mathematical framework use to model this type of problems (Richard Bellman, 1957).

Markov decision process (MDP) can define as $M = \langle S, A, R, T, \gamma, H \rangle$, when

- S is state space for optimization problem
- A is action space represent addition or changing in the solution (e.g., changing order of nodes in path)
- R is reward function mapping of states and actions. Rewards define the chosen action in particular state to improve a solution.
- T is transition function which control dynamic changing from one state to another state according to action.
- γ is scalar discount factor encouraging the agent to account for short-term rewards.
- H is horizon determining the length of the episode.

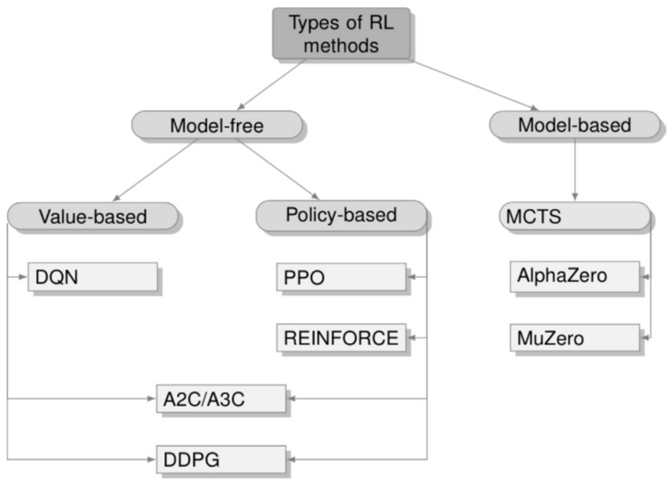
In Markov decision process (MDP), the goal of agents' action is to find a policy function $\pi(s)$ that map state into action. Solving optimization problem is finding optimal policy $\pi*$ that maximize the accumulate rewards.

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \mathbb{E}[\textstyle\sum_{t=0}^{H} \gamma^t R(s_t, a_t)] \tag{2.2}$$

For reinforcement learning algorithm that utilizes Markov decision process in order to searches for the optimal policy, it can be separated into two main categories, model-based methods, and model-free methods, (see Figure 2.13). Model-based methods normally apply in the situation where the transition functions are available or can be acquired from the environment. The agent can utilize the transition function when making decisions. Monte-Carlo tree search (MCTS) algorithm is included in this method as well. Whereas model-free methods deal with the environment without the availability of the transition function. The experiences during training process are collected in the agent's memory which can be utilized later in learning step. Moreover, model-free methods can be divided into value-based methods and policy-based methods. In case of value-based methods, a value function is approximated. A value function is used to evaluate the policy's quality for all available state-action pairs in the given environment. On the other hand, policy-based methods approximate the policy directly. In addition, there are actor–critic methods that is a combination between value-based method and policy-based method.

Figure 2.7

A Classification of Reinforcement Learning Methods (Mazyavkina et al., 2021)



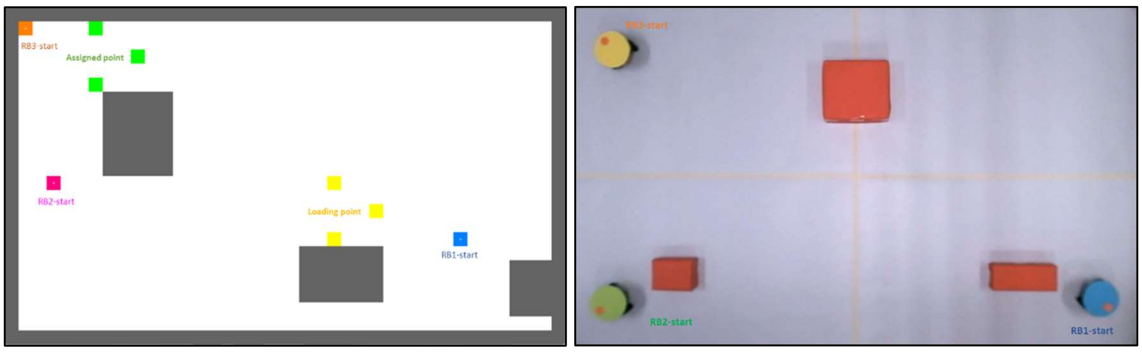
CHAPTER 3
METHODOLOGY

3.1 Overview of a System

This thesis proposed path planning based on reinforcement learning (RL) algorithm for object transportation in multi-robot system. The platform is set which area is enveloped by 1.2 x 2.0 square meter include various sizes of static obstacles (see Figure 3.1). The obstacles and robots are placed in arbitrary positions inside map's area. Loading position and delivery position are randomly assigned and be shown only in the program display. Single external camera with computer vision technique is required to obtain the information from environment by considering of the cheapest budget option.

Figure 3.1

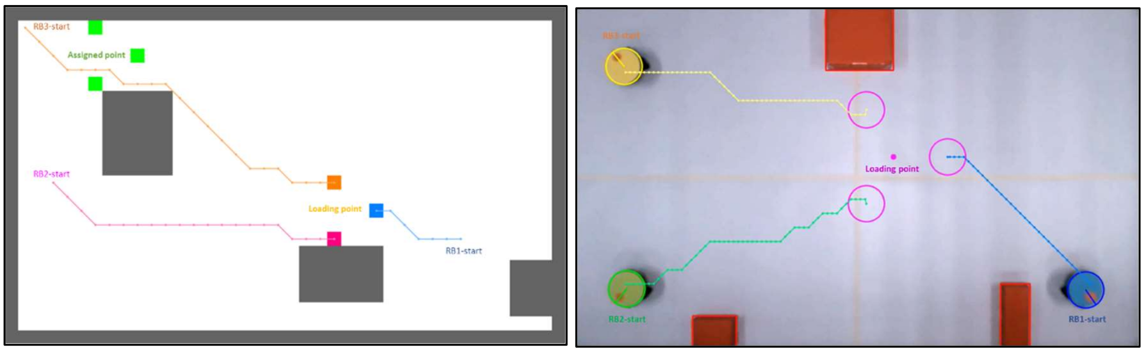
Initial State of Environment in Simulation and Experiment.



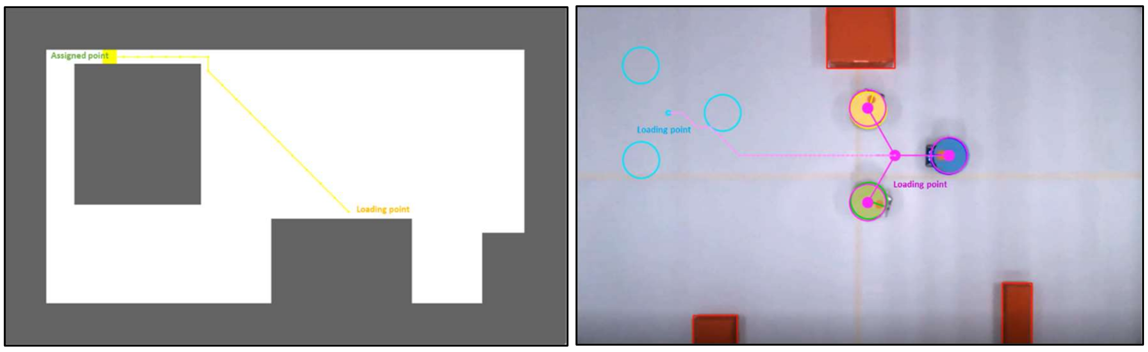
The object transportation task for multiple robots can be separated into four subtasks. First subtask is to create the optimal path with obstacle avoidance from initial position to the loading position (see Figure 3.2). In this case, just one webcam is used to obtain the information of environment. This information is transformed into observation for RL model in order to generate path for each mobile robot. For second subtask, another RL model is used to control the motion of each mobile robot to follow the planned path properly. This RL model used the distance error between robot's position and the current waypoint of planned path as an input, then RL model estimates the actions composed of left wheel speed and right wheel speed in order to control robot's movement.

Figure 3.2

Optimal Paths with Collision-Free from Reinforcement Learning Model



a) Path Planning to Loading Point

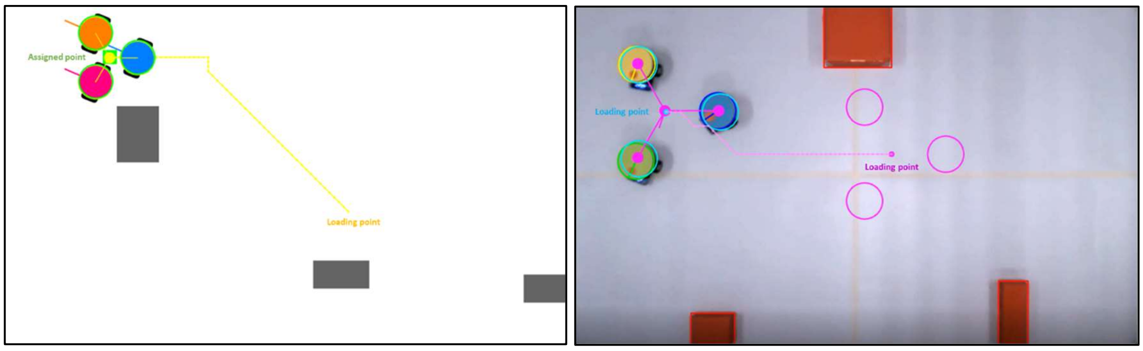


b) Path Planning to Delivery Point

After all mobile robots have reached their specified loading positions, the object will be transferred to the group of robots appropriately. The center of robot's formation is considered as a virtual position which represent as a virtual leader of a group of robots. In third subtask, the reinforcement learning for path planning takes responsibility to generate path for the virtual leader from loading position to delivery position (see Figure 3.2). Last subtask is to control all mobile robots to follow the virtual leader and maintain the formation properly until reaching the delivery position. The virtual leader is controlled by PID control in order to follow the planned path. In each time step, the reference points of all mobile robots are updated by calculating from their specified distance refer to the virtual robot. The RL model, which is trained to handle the point following task, takes responsibility to control robot's speeds by using the distance error between updated reference point and current robot position as an observation. The object transportation task will be completed when all mobile robots along with the object have reached the delivery position properly (see Figure 3.3).

Figure 3.3

Final State of Group of Mobile Robots

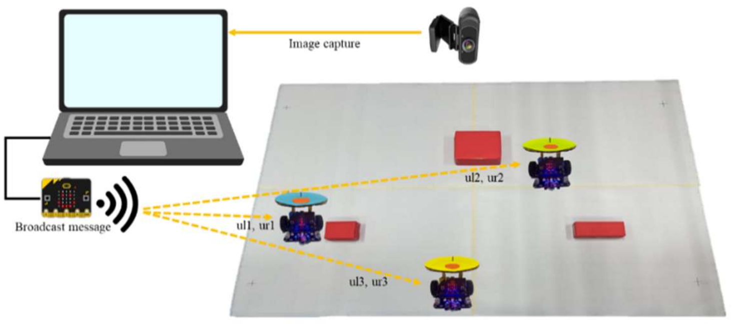


3.2 Component of a System

The platform of a system consists of three main components, computing machine, sensing equipment, and mobile robot. The sensing equipment is used to sense the information of environment. Computing machine is used to train reinforcement learning model and navigate the mobile robot. Mobile robot receives the command from the server then move accordingly to the speed in command until reaching the goal position (see Figure 3.4). In simulation, only computing machine is used for implementation. However, all components are operated together when reinforcement learning models are implemented in real environment.

Figure 3.4

Component of a System



Computing Machine

The first machine is a laptop. The specification of this laptop is an Intel Core I7-9750H CPU, a NVIDIA GeForce RTX 2060Ti GPU, 16 gigabytes of RAM, and Windows 11 OS. This laptop is used to create all programs such as RL models, environments for training and testing in simulation, program for communication between server (laptop) and client (mobile robots), and training RL models for motion control task.

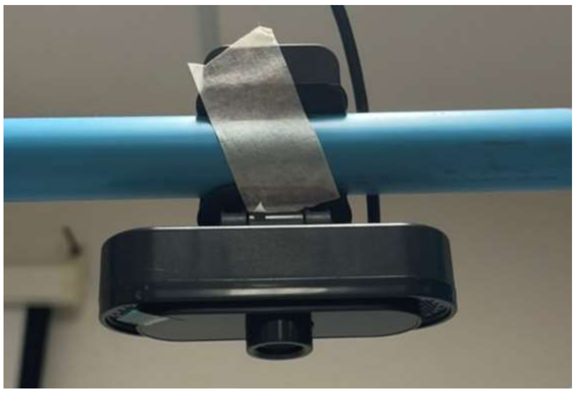
The second machine is a desktop. The specification of this desktop is an Intel Core I9-10900F CPU, a NVIDIA GeForce RTX 2080Ti GPU, 32 gigabytes of RAM, and Ubuntu 20.04 OS. This desktop is used to train the RL models for path planning with is consume large amounts of compute capability.

Sensing Equipment

Xiaovv webcam 1080P HD is used in this thesis to capture images and record videos of overall environment from the top view with image size 1080x1920 pixels (see Figure 3.5).

Figure 3.5

Xiaovv Webcam 1080P HD

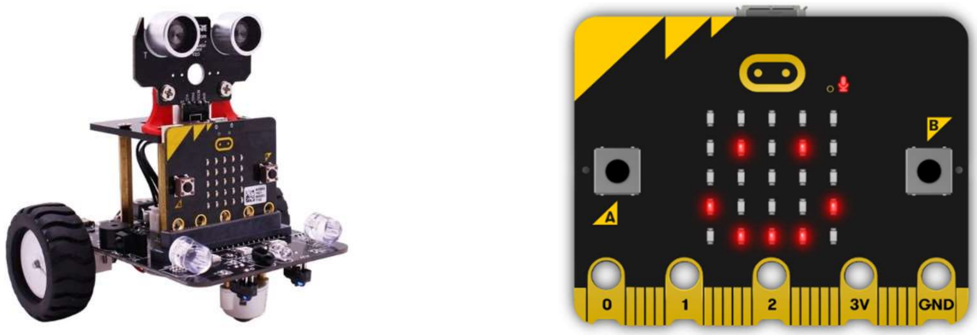


Mobile Robot

Yahboom Micro:bit smart robot car is selected to implement the RL models for object transportation in practice. Micro:bit is a build-in microcontroller for general purpose. And Yahboom smart robot car is a two wheels build-in mobile robot which controlled by Micro:bit (see Figure 3.6). This build-in robot car has a compact body and light weight. Thus, it is suitable to implement in a limited area. Micro:bit has a function to broadcast a radio message to other Micro:bit that is programed to be the same radio group. Therefore, one Micro:bit is used to receive the command speeds from the server, then broadcast to all mobile robots wirelessly in order to control the robot's speed in real time.

Figure 3.6

Yahboom Micro:bit Smart Robot Car and Micro:bit



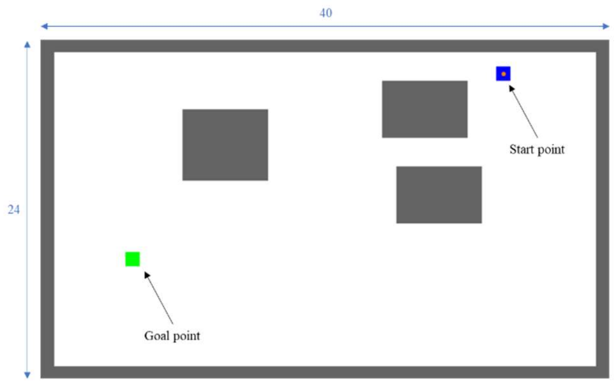
3.3 Path Planning

Path planning is the main task for this thesis. Before mobile robot can do any task such as load carriage, object rearrangement, object catching, work assembly, or map making, it must move to desired location and path planning is required firstly. There are several traditional approaches to handle with path planning. A* algorithm is one of the most popular approaches which searches for optimal available path with collision-free from start position to goal position. In this thesis, A* is used to supervise reinforcement learning model for path planning as well.

This thesis aims to create and improve reinforcement learning model which handle to path planning for any random environment optimally. Environment for training the RL model is created in python program which the agent can move in a grid map step by step. Due to the dimension of map is H1200 x W2000 square millimeter in height and width respectively. When consideration the compute capacity of available machine during training process, the map is decreased the dimension with ratio 50:1 in simulation. Thus, the map in simulation is created with sized H24 x W40 pixels. There are three static obstacles with random size between 2 to 4 pixels. They are placed in random position that doesn't overlap to each other. The initial position of robot and goal position are assigned randomly on available pixel within map's area. The distance between start position and goal position should not be lower than 32 pixel (see Figure 3.7).

Figure 3.7

Map in Simulation



When the map is created, the start and goal positions are assigned as a single point. And the obstacles are offset with the robot's radius. The different labels are assigned to each grid cell according to different object types as you can see in table 3.1. In each step, agent can move to any adjacent cell in eight directions (see Table 3.2).

Table 3.1

Label of Pixel in Grid Map

Object	Label
Free space	0
Obstacle	-1
Goal Position	1
Robot Position	2

Table 3.2

Action Space in Path Planning Respected the Image Coordinate

Action Number	Direction
0	Up-left
1	Up
2	Up-right
3	Left
4	Right
5	Down-left
6	Down
7	Down-right

The state of RL model is created by considering the current position of agent at the center. It is a cropped cells from grid map in size of H43xW75 pixels, so agent can see all obstacles and goal position wherever it is in the grid map. the pixels which are cropped outside the grid map' area is filled with label -1 same as obstacle's label. Then the cropped grid map is scaled up with ratio 1:4 and be stacked with the last four states.

Hence, agent could recognize the robot's motion in the last four steps and has a better decision-making for a next movement.

For the reward and terminal status are determined as see in table 3.3. In path planning, the agent is expected to learn how to move closer to the goal position until it can reach the goal position. The maximum length of episode is defined at 1000 step. In each step, agent learns how to find the optimal action that it could obtain maximum total reward at the end of the episode. The reward is calculated according to showing in table 3.3. It used the previous distance between robot's position and goal position subtracted by the new distance between robot' position and goal then subtract by the moving distance. If the robot moves to the direction that overlap to the static obstacle's areas, agent will be returned to its previous position and receive -3 scores as a punishment for this step. If agent can reach the goal's position, it receives 0 score and terminal status became 'True', then the episode has ended. It is obviously that the maximum reward per step is 0 score, and the minimum reward is -3 scores per step. Therefore, the main objective of this RL agent is to accomplishment the path planning with less negative score per episode as possible.

Table 3.3

Reward and Terminal Status in Path Planning

Criteria	Reward	Terminal
Reach the goal	0	True
Hit obstacle	-3	False
Other	R	False

Where $R = (\Delta s_{prev} - \Delta s_{current}) - \Delta d$

Δs_{prev}	is a previous distance between robot's position and goal
$\Delta s_{current}$	is a current distance between robot's position and goal
Δd	is a moving distance in this step

Due to the obstacles, start position, and goal position are randomly generated in each episode independently, so it is hard to compare the reward in episode by episode directly. Moving average reward of last 100 episodes is applied to find the model's parameters during a training period. To reach this thesis' objective in order to find the reinforcement learning model for path planning, there are three steps for the experiment as list below.

1. Evaluate convolutional layers.
2. Evaluate reinforcement learning algorithms.
3. Explore an effect of semi-supervised with A* algorithm to reinforcement learning model.

Deep Q-Learning (DQN)

To test the first assumption above, deep Q-learning network is selected as an initial reinforcement learning model because DQN is one of the most popular and well-known RL models to handle with the environment that has discreate actions. DQN is improved from Q-learning algorithms by use a neural network to approximate the Q-value function instead of using Q-table (Huang, Y. 2020). The state is given as the input in the neural network and Q-value of all possible actions is generated as an output. The state generate in this simulation is grid map sized H24xW40 pixels, and the state is changed in every step respect to the current position of an agent. Thus, it is too difficult to limit the number of states in Q-table with Q-learning techniques. DQN could solve this problem by using convolution network to generate the feature map representing the information of the state in each step then feed to neural networks with ReLU activation function to approximate all possible Q-values. This DQN model select the action respect to epsilon greedy value. The epsilon value starts from 1 and decreasing every step by decay parameter. In every step, the model generates a random float from 0 to 1, if the random float more than current epsilon value, model send the random action. In the other hand, if the random float less than current epsilon value, the model provides the action which has maximum Q-value. During the training process, the network's parameters are updated by mean square error (MSE) between Q-value from the current model and the target Q-value. The pseudo-code of DQN model is shown in algorithm1.

Algorithm 1 Deep Q-learning with experience replay

```
Initialize experience replay memory $D$ to capacity $K$
Initialize network $Q$ with random weights
Initialize the Agent to interact with the environment
**for** episode = 1, $M$ **do**
    $\epsilon \leftarrow$ setting new epsilon with $\epsilon$-decay
    With probability $\epsilon$ select the random action $a_t$
    Otherwise select $a_t = max_a Q(s_t, a;\ \theta)$
    Agent takes action $a_t$ and observe reward $r_t$ and state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1}, terminal)$ in $D$
    **If** enough experience in $D$ **then**
        Sample random minibatch $N$ of transitions $(s_t, a_t, r_t, s_{t+1}, terminal)$ from $D$
        Compute target Q-value:
        $Q^* = r_t + \gamma max_{a'} Q(s_{t+1}, a';\ \theta)(1 - terminal)$
        Calculate the loss $\mathcal{L} = \frac{1}{N} \sum_{i=0}^{N-1} (Q(s_i, a_i) - Q^*)^2$
        Update $Q$ using a gradient descent algorithm by minimize the loss $\mathcal{L}$
    Set $s_t \leftarrow s_{t+1}$
    end
end
```

The hyperparameters of DQN approach which are determined for training process are shown in table 3.4.

Table 3.4

The Hyperparameters of DQN Approach for Path Planning

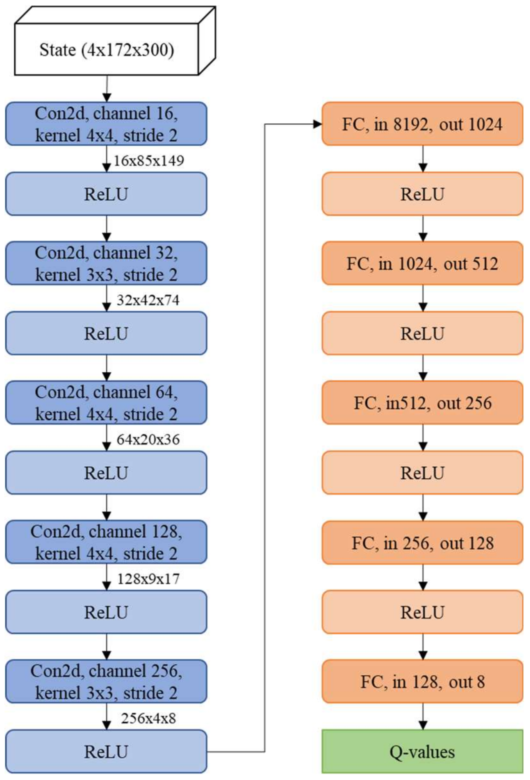
hyperparameter	Setting Value
Episode number	10,000
Max step	1,000
Learning rate (α)	0.001
Discount factor (γ)	0.99
ϵ-start	1.0
ϵ-min	0.01
ϵ-decay	0.99999
Memory capacity (K)	5000
Minibatch (N)	64

Convolution neural networks (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. The layers of a CNN consist of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers and normalization layers. The removal of limitations and increase in efficiency for image processing results in a system that is far more effective, simpler to trains limited for image processing. In order to find an appropriate convolutional layer for this thesis' state, three different convolutional layers are selected to test the performance consisted of ordinary convolutional layer, ResNet18, and ResNet50.

Ordinary Convolutional Network

Figure 3.8

Structure of Ordinary Convolution Network for Path Planning


State (4x172x300)
Con2d, channel 16, kernel 4x4, stride 2
16x85x149
ReLU
Con2d, channel 32, kernel 3x3, stride 2
32x42x74
ReLU
Con2d, channel 64, kernel 4x4, stride 2
64x20x36
ReLU
Con2d, channel 128, kernel 4x4, stride 2
128x9x17
ReLU
Con2d, channel 256, kernel 3x3, stride 2
256x4x8
ReLU
FC, in 8192, out 1024
ReLU
FC, in 1024, out 512
ReLU
FC, in512, out 256
ReLU
FC, in 256, out 128
ReLU
FC, in 128, out 8
Q-values

Ordinary Convolution Network of this thesis consists of five convolutional layers with ReLU activation function and four fully connected layers with ReLU activation function. The last layer is a linear layer which approximates the Q-values (see Figure 3.8). The network takes state sized 4x172x300 as an input, the convolutional layer transforms the input into a feature map then feed to fully connected layer. Finally, the network generates eight Q-values.

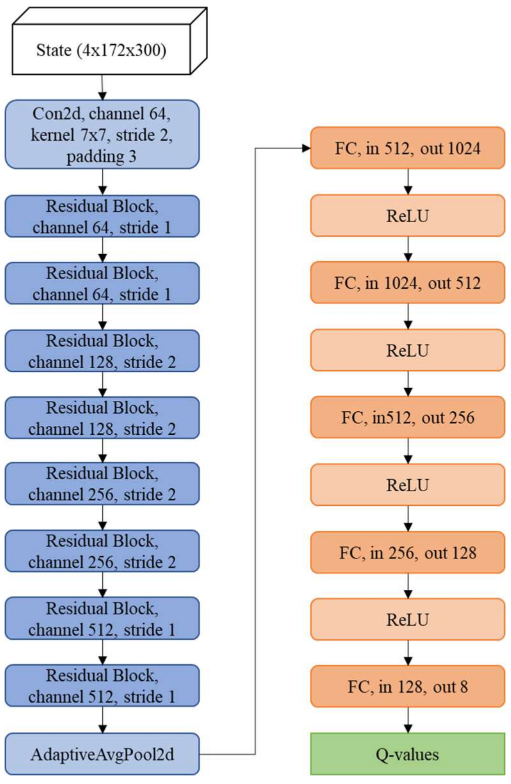
ResNet

The ResNet Convolution Neural Network was first described in the ResNet paper (He et al., 2016). It is the popular as a baseline model today. Before the rise of ResNet, there were many difficulties when building very deep neural networks. Normalization technique could solve the problem when increasing the network depth. However, when the depth goes beyond 10 – 20 layers, the problem of degradation has occurred. With additional network depth, training error has decreased to some point then it starts to increase. ResNet introduced a mechanism for shortcut connections that prevents the scrambling of effect of newly initialized layers by bypassing them, allowing undegraded gradient information to be efficiently backpropagated to the earliest layers of the model.

ResNet18 is the simplest ResNet with 8 residual blocks (there are two convolutional layers in each residual block), the initial convolution layer, and the final linear layer with sofmax activation function, giving a total 18 layers. However, ResNet18 was modified in this thesis by replacing the last fully connected layer with five fully connected layers with ReLU activation function as shown in Figure 3.9.

Figure 3.9

Structure of Modified ResNet18 for Path Planning


State (4x172x300)
Con2d, channel 64, kernel 7x7, stride 2, padding 3
Residual Block, channel 64, stride 1
Residual Block, channel 64, stride 1
Residual Block, channel 128, stride 2
Residual Block, channel 128, stride 2
Residual Block, channel 256, stride 2
Residual Block, channel 256, stride 2
Residual Block, channel 512, stride 1
Residual Block, channel 512, stride 1
AdaptiveAvgPool2d
FC, in 512, out 1024
ReLU
FC, in 1024, out 512
ReLU
FC, in512, out 256
ReLU
FC, in 256, out 128
ReLU
FC, in 128, out 8
Q-values

The residual block is a reusable block. ResNet18 and ResNet34 use a basic residual block, but ResNet50, ResNet101, and ResNet152 use bottleneck block which is more complicated residual blocks with three convolutions. We also need two types of residual block, one that preserves feature map size and one that allows changes to the feature map size (see Figure 3.10 & 3.11).

Figure 3.10

Structure of Residual Blocks, Preserved and Not Preserved Feature Map Size

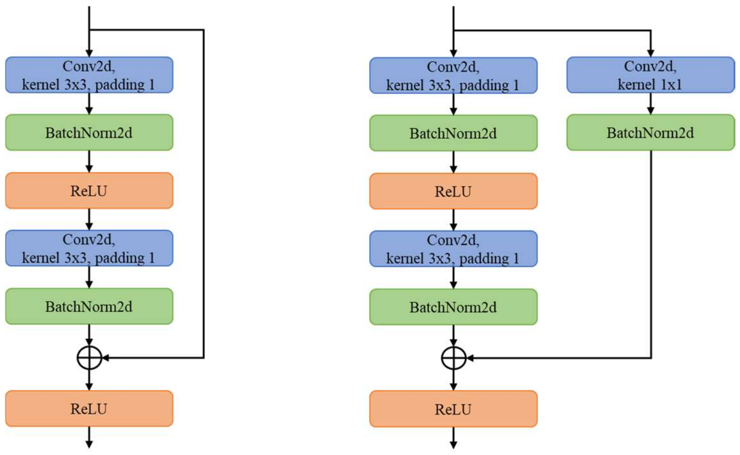
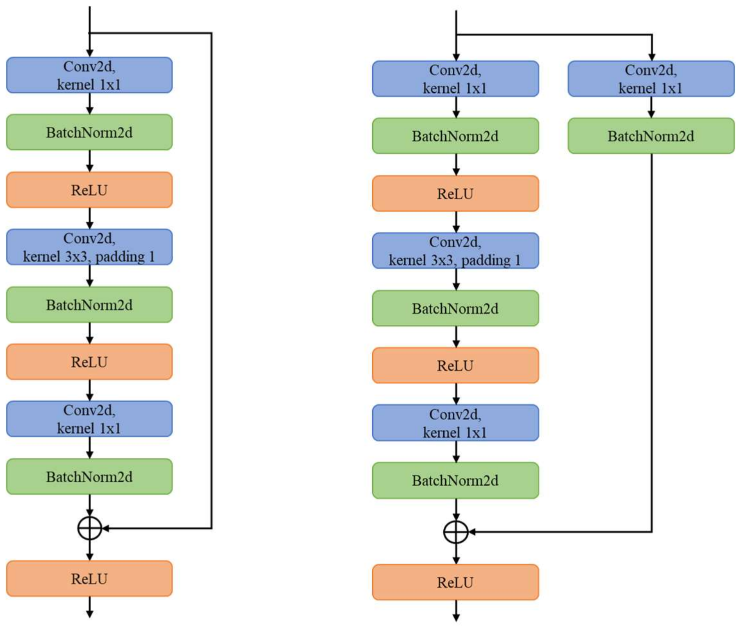

Conv2d, kernel 3x3, padding 1
BatchNorm2d
ReLU
Conv2d, kernel 3x3, padding 1
BatchNorm2d
ReLU
Conv2d, kernel 3x3, padding 1
BatchNorm2d
ReLU
Conv2d, kernel 3x3, padding 1
BatchNorm2d
ReLU
Conv2d, kernel 1x1
BatchNorm2d

Figure 3.11

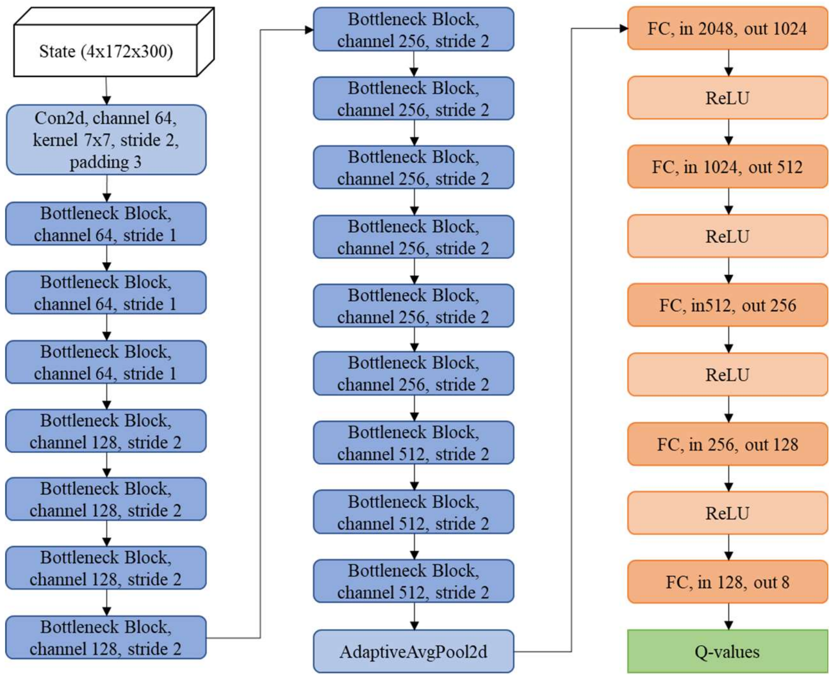
Structure of Bottleneck Blocks, Preserved and Not Preserved Feature Map Size



ResNet50 which is more complicated when compared to ResNet18 consists of 16 bottleneck blocks (there are three convolutional layers in each bottleneck block), the initial convolution layer, and the final linear layer with sofmax activation function, giving a total 50 layers. However, ResNet50 was modified in this thesis by replacing the last fully connected layer same structure as modification in ResNet18 as shown in Figure 3.12.

Figure 3.12

Structure of Modified ResNet50 for Path Planning


State (4x172x300)
Con2d, channel 64, kernel 7x7, stride 2, padding 3
Bottleneck Block, channel 64, stride 1
Bottleneck Block, channel 64, stride 1
Bottleneck Block, channel 64, stride 1
Bottleneck Block, channel 128, stride 2
Bottleneck Block, channel 128, stride 2
Bottleneck Block, channel 128, stride 2
Bottleneck Block, channel 128, stride 2
Bottleneck Block, channel 256, stride 2
Bottleneck Block, channel 256, stride 2
Bottleneck Block, channel 256, stride 2
Bottleneck Block, channel 256, stride 2
Bottleneck Block, channel 256, stride 2
Bottleneck Block, channel 256, stride 2
Bottleneck Block, channel 512, stride 2
Bottleneck Block, channel 512, stride 2
Bottleneck Block, channel 512, stride 2
AdaptiveAvgPool2d
FC, in 2048, out 1024
ReLU
FC, in 1024, out 512
ReLU
FC, in512, out 256
ReLU
FC, in 256, out 128
ReLU
FC, in 128, out 8
Q-values

Three different structure of convolution network are trained individually. The results are compared to find which convolution network has the best performance in order to find the optimal path when encounter with the random maps. After finished training process, the network with parameter that has the best reward during training was selected. All three convolutional network were evaluated by running 100 episodes in simulation. The convolution network that has the highest episode reward will be select as a winner. The results of the experiment are shown in Evaluation session.

Next assumption is to evaluate reinforcement learning algorithm for path planning problem. When consideration of the state as referred in previous, there is unlimited number of possible states which can generate from the environment. This type of countless number of states is suitable for reinforcement learning with model-free method to handle with. In this thesis, there are three reinforcement learning algorithms were selected for comparison the performance in path planning problem those are deep Q-learning network (DQN), double deep Q-learning network (DDQN), and proximal policy optimization (PPO).

DQN and DDQN are reinforcement learning algorithm in value-based method that find the optimal policy via approximation Q-value. And PPO is reinforcement learning algorithm in policy-based method that directly find the optimal policy via optimization the policy parameters. Due to DQN was already mentioned previously, thus only DDQN and PPO algorithm will be described in this part.

Double Deep Q-Learning Network (DDQN)

Double deep Q-learning network is developed in order to solve the problem of overestimation of action-value in deep Q-learning network (Xiao et al., 2020). When we consider target Q-value in DQN algorithm, the term $max_{a'}Q(s_{t+1}, a')$ take the maximum overestimate value that led to maximization bias in learning process. This problem leads to unstable training and low-quality policy. This problem will be solved by using DDQN. The solution involves using two separated Q networks, primary network and target network, each network is used to update each other. The primary Q network used to estimate Q-value of current state and was updated in every step. On the other hand, target network used to estimate the Q-values of next state and was updated in particular time duration by copy the primary network's parameters. The calculation of target Q-value in DDQN algorithm and learning step are shown in algorithm 2.

Algorithm 2 Double deep Q-learning with experience replay

```
Initialize experience replay memory $D$ to capacity $K$
Initialize network $Q_\theta$ and target network $Q_{\theta'}$ with random weights
Update target network parameters $\theta' \leftarrow \theta$
```

```
Initialize the Agent to interact with the environment
for episode = 1, M do
    $\epsilon \leftarrow$ setting new epsilon with $\epsilon$-decay
    With probability $\epsilon$ select the random action $a_t$
    Otherwise select $a_t = max_a Q_\theta(s_t, a)$
    Agent takes action $a_t$ and observe reward $r_t$ and state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1}, terminal)$ in D
    If enough experience in D then
        Sample random minibatch N of transitions $(s_t, a_t, r_t, s_{t+1}, terminal)$ from D
        Compute target Q-value:
        $Q^* = r_t + \gamma Q_{\theta'}(s_{t+1}, argmax_{a'} Q_\theta(s_{t+1}, a'))(1 - terminal)$
        Calculate the loss $\mathcal{L} = \frac{1}{N}\sum_{i=0}^{N-1}(Q(s_i, a_i) - Q^*)^2$
        Update $Q_\theta$ using a gradient descent algorithm by minimize the loss $\mathcal{L}$
        If reach the period of updating target network, $Q_{\theta'}$ then
            Update target network parameters $\theta' \leftarrow \theta$
    Set $s_t \leftarrow s_{t+1}$
    end
end
```

The difference between DQN and DDQN is shown in the derived equation of target Q-value, especially in term of maximum Q-value of next state. In DQN algorithm, it calculates the maximum Q-value of the next state using primary network Q_θ which is always changing during training time. Whereas DDQN calculates the maximum Q-value of the next state using target network $Q_{\theta'}$ but the next action is selected from the highest Q-value of next state by primary network Q_θ. The target network's parameters are fixed in particular duration and update to be the same as the present primary network's parameters. With this technique, the network Q should be improved the performance in convergence compared to ordinary DQN. The hyperparameters of DDQN approach which are determined for training process are shown in table 3.5.

Table 3.5

The Hyperparameters of DDQN Approach for Path Planning

hyperparameter	Setting Value
Episode number	10,000
Max step	1,000
Learning rate (α)	0.001
Discount factor (γ)	0.99
ϵ-start	1.0
ϵ-min	0.01
ϵ-decay	0.99999
Memory capacity (K)	5000
Minibatch (N)	16
Target $Q_{\theta'}$ update period	2,000

Proximal Policy Optimization (PPO)

Proximal policy optimization is an interesting algorithm in the field of reinforcement learning, which provides an improvement on trust region policy optimization (TRPO). This algorithm was proposed in 2017 (Schulman et al. 2017). PPO is a policy gradient method and can be used for environments with either discrete or continuous action spaces. It trains a stochastic policy in an on-policy way and utilizes the actor critic method. The actor maps the observation to an action and the critic gives an expectation of the rewards of the agent for the given observation. Firstly, a set of trajectories consist of state, action, $\log \pi_\theta\ (a_t|s_t)$, reward, next state, and terminal of each step is collected in a memory. When the memory is filled, it collects a set of trajectories for each epoch by sampling from the memory and the advantage estimates are computed in order to update the policy and fit the value function. The policy is updated via a stochastic gradient ascent optimizer, while the value function is fitted via some gradient descent algorithm. This procedure is applied for many epochs until the environment is solved. The pseudocode representing the learning step of PPO is shown in algorithm 3.

Algorithm 3 Proximal policy optimization with clipped surrogate objective

```
Initialize memory to collect trajectories $D$ with capacity $K$
Initialize policy parameters $\theta$
Initialize value function parameters $\phi$
Initialize the Agent to interact with the environment
for episode = 1, $M$ do
    Select action $a_t$ by policy $\pi_\theta$
    Agent runs action $a_t$ and observe reward $r_t$ and state $s_{t+1}$
    Collect set of trajectories $(s_t, a_t,\ \log\pi_\theta\ (a_t|s_t),\ r_t, s_{t+1}, terminal)$ in $D$
    If memory $D$ is filled with capacity $K$ then
        Compute expected reward, $\hat{R}_t = r_t + \gamma V_\phi(s_{t+1})(1 - terminal)$
        Compute advantage estimates, $\hat{A}_t = \hat{R}_t - V_\phi(s_t)$
        for epoch = 1, $P$ do
            Sample minibatch $N$ of trajectories from memory until empty
            Calculate probability ratio, $r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$
            Calculate policy loss with clipped surrogate objective:
            $L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ clip(r_t(\theta),\ 1 - \epsilon, 1 + \epsilon)\hat{A}_t\right)\right]$
            Calculate value function loss, $L(\phi) = \frac{1}{N}\sum_{i=0}^{N-1}(V_\phi(s_t) - \hat{R}_t)^2$
            Total loss, $L_{tot} = L^{CLIP}(\theta) + 2L(\phi)$
            Update policy parameter $\theta$ and value function parameters $\phi$
            using a gradient descent algorithm
  Set $s_t \leftarrow s_{t+1}$
    end
end
```

In this thesis, the policy network and value function network are combined to be single network (see Figure 3.13). The network takes observation as input then estimate state value and generate action distribution as the same time. The total loss to update network is shown in algorithm 3 and was update with Adam optimizer. The last layer of actor applying Softmax activation function to estimate probability for each action then the action was sampled via this probability. The hyperparameters of PPO approach which are determined for training process are shown in table 3.6.

Figure 3.13

Structure of PPO for Path Planning

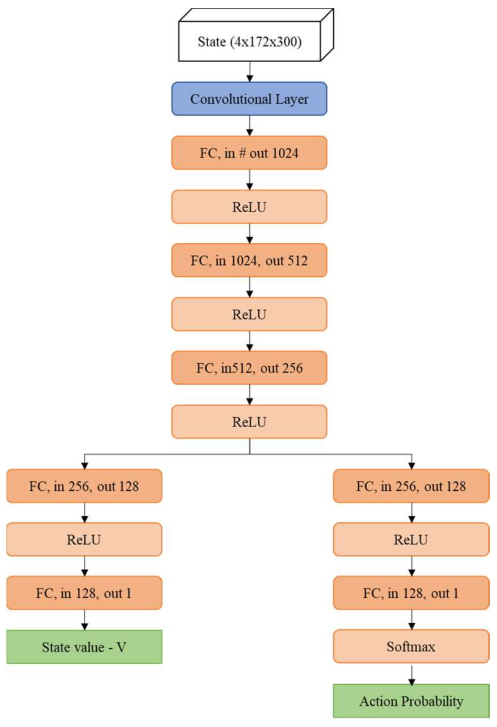


Table 3.6

The Hyperparameters of PPO Approach for Path Planning

hyperparameter	Setting Value
Episode number	10,000
Max step	1,000
Learning rate (α)	0.001
Discount factor (γ)	0.99
ϵ-clipped	0.1
Number of Epoch (P)	30
Memory capacity (K)	256
Minibatch (N)	32

In the last step to find reinforcement learning model for path planning, the best model will be supervised by the best well-known path planning algorithm, A*. A* algorithm guarantees to find the optimal path in this grid world environment. The experiences played with A* are collected in replay buffer memory for exploitation. There is mixture experience between exploration from RL policy π, and exploitation from A*. Hence, in learning step, the RL model supposes to learn for finding optimal path as well. Moreover, this experiment would like to see the appropriate proportion of experience replays which are collected via RL model and A* algorithm. the assumption is tested with three scenarios, no supervised, 25% supervised with A*, and 50% supervised with A*. The model after training should find the optimal path in any environment and training time will be considered as well.

3.4 Motion Control Task

After path was created from current position to desire goal, motion control part is required to allow mobile robot moving to the target location. In this thesis, differential wheeled mobile robot is selected to implement the object transportation in practice due to it is easy to build and control. Differential wheeled mobile robot is nonholonomic robot which its controllable degree of freedom is less than the total degrees of freedom. A car has three degrees of freedom, i.e., its position in two axes and its orientation. However, there are only two controllable degrees of freedom which are left wheel speed and right wheel speed. This makes it difficult to turn the car in any direction unless the car slides. In addition, it is more challenge to form the formation and transport the object using nonholonomic mobile robots.

Equation 3.1 to 3.2 shows the relation between robot's linear velocity and angular velocity when left wheel's speed and right wheel's speed are provided.

$$v = \frac{(v_l + v_r)}{2},\ \Delta s = vdt \tag{3.1}$$

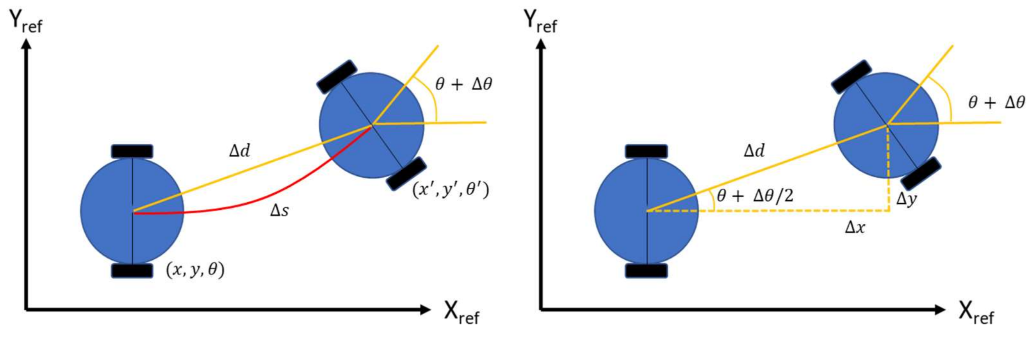
$$\omega = \frac{(v_l - v_r)}{l},\ \Delta\theta = \omega dt \tag{3.2}$$

Where v is a linear velocity, Δs is displacement, ω is an angular velocity, $\Delta\theta$ is changing of heading, v_l is left wheel's speed, v_r is right wheel's speed, l is the distance between two wheels, and dt is time period.

In figure 3.14, assume robot is at some position (x, y) in reference coordinate, headed in a direction making an angle θ with X axis. We assume the robot's center is at the midpoint along the wheel axle. By manipulating the control parameters v_l, v_r, the displacement Δs and changing of orientation $\Delta\theta$ can be calculated by equation 3.1. Thus, we can get the robot to move to different positions (x', y') and orientations θ'.

Figure 3.14

Displacement of Two Wheels Mobile Robot



Due to the displacement Δs occurred within short period is small, so it can be assumed $\Delta d = \Delta s$, then we can find the displacement in x-axis, the displacement in y-axis, and changing of heading using equation 3.3 to 3.4. Finally, the new position (x', y') and orientations θ' can be obtained from equation 3.5.

$$\Delta x = \Delta s \cos(\theta + \Delta\theta/2) \tag{3.3}$$

$$\Delta y = \Delta s \sin(\theta + \Delta\theta/2) \tag{3.4}$$

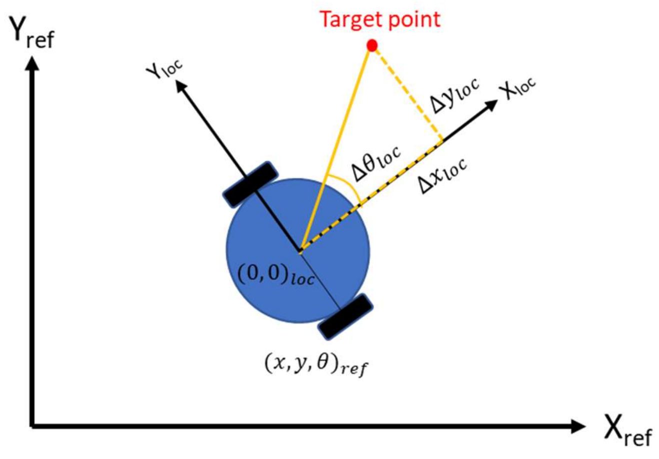
$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} vdt\cos(\theta + \omega dt/2) \\ vdt\sin(\theta + \omega dt/2), \\ \omega dt \end{bmatrix} \tag{3.5}$$

According to equation 3.5, to display the mobile robot's motion in simulation, it used only linear velocity and angular velocity with dt to update the robot's position. However, in practice, we need to transform linear velocity and angular velocity to left wheel speed and right wheel speed then sending the command to mobile robots.

To control the robot's motion, the error between the robot's current position and target position is used in order to generate the outputs as linear velocity and angular velocity. The robot's local coordinate was set up considering robot's position at point (0, 0), as you can see in figure 3.15. The target position was transformed into robot's coordinate then error in x-axis Δx_{loc} is used to generate linear velocity v and error in orientation $\Delta\theta_{loc}$ is used to generate angular velocity ω.

Figure 3.15

Errors in Robot's Coordinate

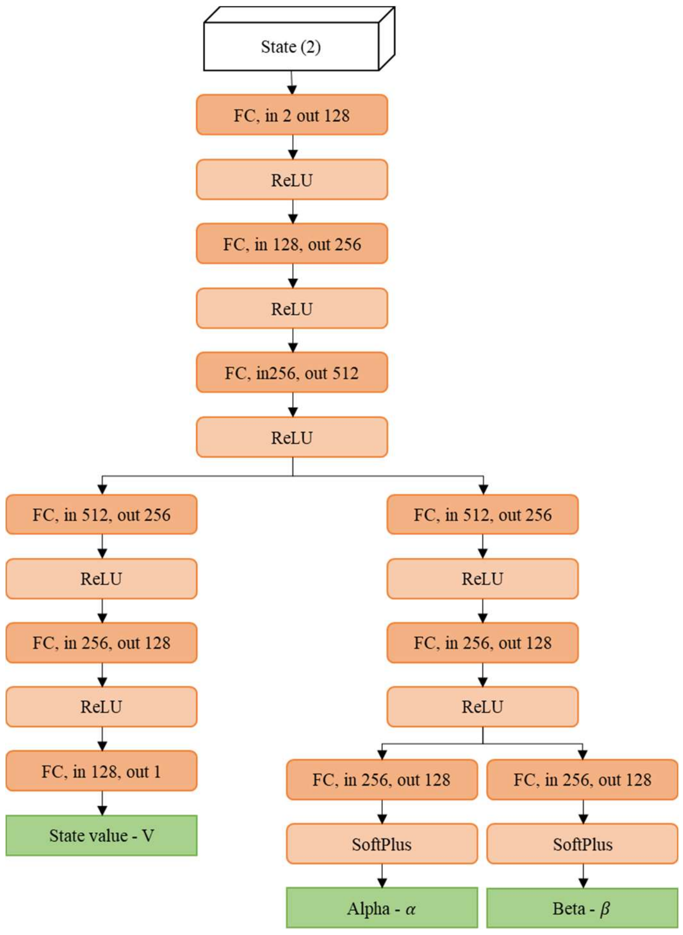


Policy-based reinforcement learning method is suitable to deal with the problem with continuous action space. In this thesis, there are three reinforcement learning algorithms which are selected to perform motion control task, proximal policy optimization (PPO), deep deterministic policy gradient (DDPG), and soft actor-critic (SAC).

As mentioned previously, proximal policy optimization is one of Policy Gradient method that can be used for environments with either discrete or continuous action spaces by training a stochastic policy and utilizing the actor critic method. The network for the motion control task is shown in figure 3.16.

Figure 3.16

Structure of PPO for Motion Control Task



According to the PPO network, the input is an observation which consist of two errors, distance error Δx and heading error $\Delta\theta$. It has two separate branches in this network, one called value function used to estimate state value and other called policy network used to generate actions in form of linear velocity v and angular velocity ω. Policy network generates two values for each action, Alpha-α and Beta-β then utilize the function in pytorch called “Beta”. Beta function is one of distribution in exponential family which require two attributes, concentration1 and concentration2. Then the action is sampled from this distribution. Training step of this PPO network still follows

Algorithm 3. And the hyperparameters for training process of this PPO network are determined as shown in table 3.7.

Table 3.7

The Hyperparameters of PPO Approach for Motion Control Task

hyperparameter	Setting Value
Episode number	500
Max step	400
Learning rate (α)	0.001
Discount factor (γ)	0.99
ϵ-clipped	0.1
Number of Epoch (P)	20
Memory capacity (K)	1024
Minibatch (N)	128

Deep Deterministic Policy Gradient (DDPG)

Deep deterministic policy gradient is an algorithm which simultaneously learns a Q-function and a policy. It uses off-policy data and bellman equation to learn the Q-function and uses the Q-function to learn the policy (Lillicrap et al., 2016). This approach is closely related to deep Q-learning. Instead of learning an approximator to $Q^*(s, a)$, DDPG learns an approximator to $a^*(s)$ and it can handle any specifically adapted environment with continuous action spaces. When there is an infinite number of action spaces, we can't compute all Q-values and find the best action which has the maximum Q-value among them. But DDPG maps the any given environment to continuous action spaces directly with the policy. And it has a Q-function to evaluate how good of that action is. The learning step of DDPQ is shown in Algorithm 4.

Algorithm 4 Deep deterministic policy gradient

```
Initialize experience replay memory $D$ to capacity $K$
Initialize policy $\mu_\theta$, Q-function $Q_\phi$, target policy $\mu_{\theta'}$, target Q-function $Q_{\phi'}$
Set target policy parameters $\theta' \leftarrow \theta$, and target Q-function parameters $\phi' \leftarrow \phi$
```

```
Initialize the Agent to interact with the environment
for episode = 1, $M$ do
    Select $a_t = \mu_\theta(s_t)$
    Agent executes action $a_t$ and observe reward $r_t$ and state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1}, terminal)$ in $D$
    If enough experience in $D$ then
        Sample random minibatch $N$ of transitions $(s_t, a_t, r_t, s_{t+1}, terminal)$ from $D$
        Compute target Q-value:
        $Q^* = r_t + \gamma Q_{\phi'}(s_{t+1}, \mu_{\theta'}(s_{t+1}))(1 - terminal)$
        Compute the MSE loss, $L_\phi = \frac{1}{N}\sum_{i=0}^{N-1}(Q_\phi(s_i, a_i) - Q^*)^2$
        Update $Q_\phi$ using a gradient descent algorithm
        Compute the policy loss, $L_\theta = \frac{1}{N}\sum_{i=0}^{N-1} Q_\phi\big(s_i, \mu_\theta(s_i)\big)$
        Update $\mu_\theta$ using a gradient ascent algorithm
        Update target networks' parameters with:
        $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$
        $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
    Set $s_t \leftarrow s_{t+1}$
    end
end
```

Likely to double deep Q-learning, in DDPG algorithm, there are target Q-function and target policy used to compute target Q-value then updating Q-function Q_ϕ using a gradient descent algorithm. The policy μ_θ is updated by using updated Q-function with a gradient ascent algorithm. Later, each learning step, the target policy $\mu_{\theta'}$ and target Q-function $Q_{\phi'}$ are updated by cloning the current network parameters with small proportion τ. The figure 3.17 shows the structures of policy network and Q-function network. It used a simple structure of neural network for Q-function with fully connected layers and ReLU. For policy network, the last layer applied Tanh activation function in order to limit the boundary of action within specific range. The hyperparameters for training process of DDPG approach are determined as shown in table 3.8

Figure 3.17

Structures of Policy and Q-Function of DDPG for Motion Control Task

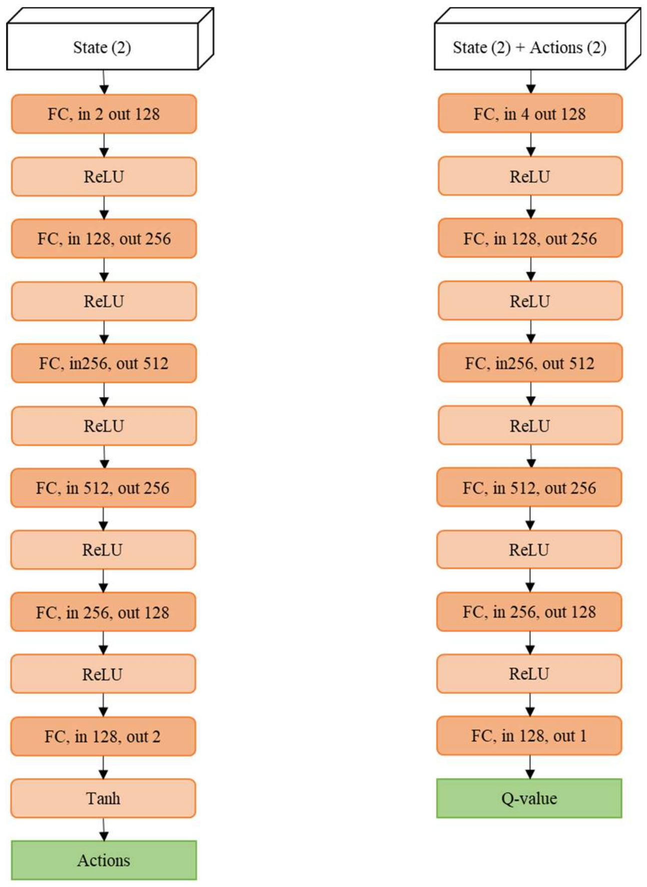


Table 3.8

The Hyperparameters of DDGP Approach for Motion Control Task

Hyperparameter	Setting Value
Episode number	500
Max step	400
Learning rate of policy network (α)	0.0005
Learning rate of Q-function network (β)	0.001
Discount factor (γ)	0.99
Tau (τ)	0.001
Memory capacity (K)	10000
Minibatch (N)	128

Soft Actor-Critic (SAC)

Soft actor-critic is an algorithm that optimizes a stochastic policy in an off-policy way. It is a combination between stochastic policy optimization and DDPG approach (Haarnoja et al., 2018). A main feature of SAC is entropy regularization. The policy is trained to maximize a trade-off between expected return and entropy which measures a randomness in the policy. It is similar to the exploration-exploitation trade-off. An increasing entropy results in more exploration, which can accelerate a learning later. It can also prevent the policy from too early converging to a bad local optimum. SAC simultaneously learns a policy π_θ, two Q-functions Q_{ϕ_1}, Q_{ϕ_2}, and value-function V_ψ as be shown in Algorithm 5.

Algorithm 4 Soft actor-critic

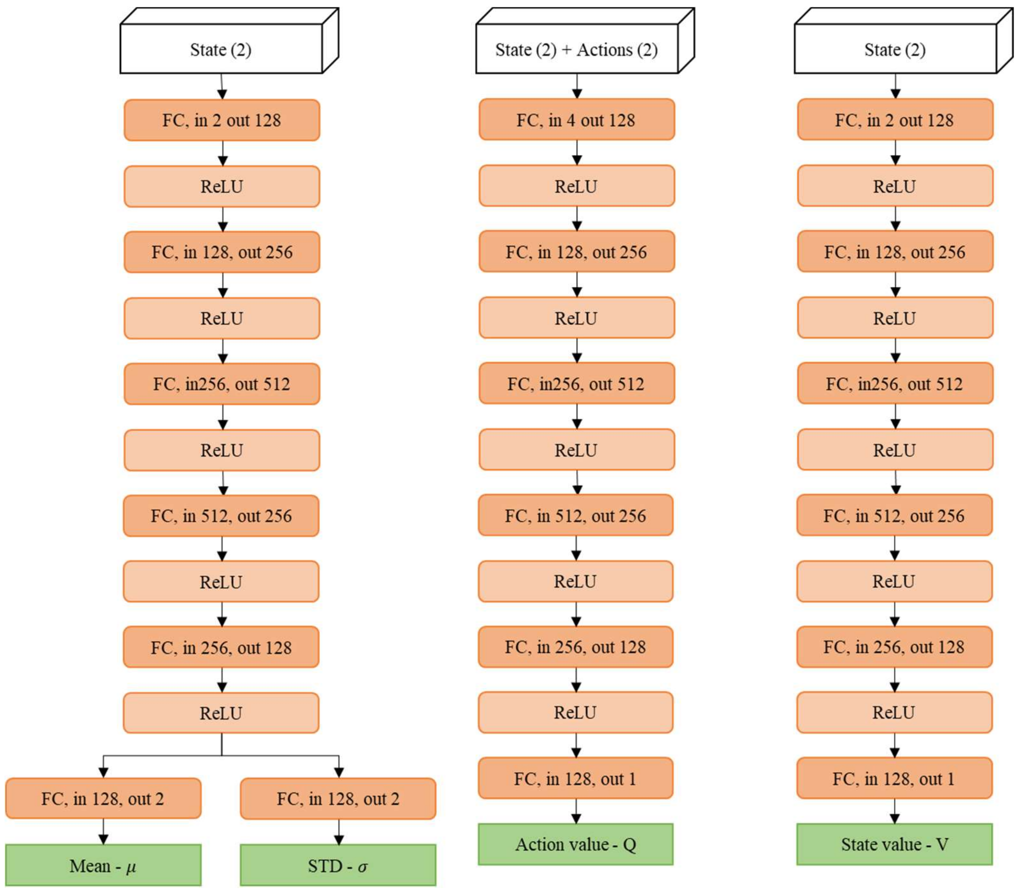
```
Initialize experience replay memory D to capacity K
Initialize policy $\pi_\theta$, two Q-functions $Q_{\phi1}, Q_{\phi2}$, value-function $V_\psi$ and target value-
function $V_{\psi'}$
Set target value-function parameters $\psi' \leftarrow \psi$
Initialize the Agent to interact with the environment
for episode = 1, M do
    Select $a_t = \pi_\theta(a|s_t)$
    Agent executes action $a_t$ and observe reward $r_t$ and state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1}, terminal)$ in D
    If enough experience in D then
        Sample random minibatch N of transitions $(s_t, a_t, r_t, s_{t+1}, terminal)$ from D
        Compute target value:
        $V^* = \min\left(Q_{\phi1}(s_t, \tilde{a}),\ Q_{\phi2}(s_t, \tilde{a})\right) - \log \pi_\theta(\tilde{a}|s_t), \tilde{a} \sim \pi_\theta(a|s_t)$
        Compute the value loss, $L_\psi = \frac{1}{N}\sum_{i=0}^{N-1}(V_\psi(s_i) - V^*)^2$
        Update $V_\psi$ using a gradient descent algorithm
        Compute the policy loss:
        $L_\theta = \frac{1}{N}\sum_{i=0}^{N-1}(\min\left(Q_{\phi1}(s_t, \tilde{a}),\ Q_{\phi2}(s_t, \tilde{a})\right) - \log \pi_\theta(\tilde{a}|s_t))$
        Update $\pi_\theta$ using a gradient ascent algorithm
        Compute target Q-value:
```

```
        $Q^* = r_t + \gamma(V_{\psi'}(s_{t+1}))(1 - terminal)$
        Compute Q-value loss:
        $L_{\phi 1} = \left(\frac{1}{N}\sum_{i=0}^{N-1}\left(Q_{\phi 1}(s_i, a_i) - Q^*\right)^2\right), L_{\phi 2} = \left(\frac{1}{N}\sum_{i=0}^{N-1}\left(Q_{\phi 2}(s_i, a_i) - Q^*\right)^2\right)$
        $L_{\phi tot} = (L_{\phi 1} + L_{\phi 2})/2$
        Update $Q_{\phi 1}, Q_{\phi 2}$ using a gradient descent algorithm by minimize the loss $L_{\phi tot}$
        Update target value-function with
        $\psi' \leftarrow \tau\psi + (1 - \tau)\psi'$
    Set $s_t \leftarrow s_{t+1}$
    end
end
```

There are totally five networks in this SAC approach, policy π_θ, two Q-function $Q_{\phi 1}, Q_{\phi 2}$, value function V_ψ, and target value function $V_{\psi'}$. Two Q-functions and policy are used to compute target value. The value function is updated using gradient descent algorithm with MSE loss between value function and this target value. The policy loss is computed by using two Q-functions and current policy with reparameterization trick. The policy is updated using gradient assent algorithm. For the updating of Q-function, it uses a gradient descent algorithm to minimize the average loss of two Q-functions and update both networks at the same time. The structure of three networks, policy, Q-function, and value function are shown in figure 3.18.

Figure 3.18

Structures of Policy, Q-Function, and V-Function of SAC for Motion Control Task



The policy network has five layers of fully connected layers with ReLU activation function. There are two parallel layers at the last layer to estimate mean-μ and std-σ for each action. The networks of Q-function and value function are consisted of six fully connected layers with ReLU to estimate action value and state value. The hyperparameters of SAC approach in training process are shown in table 3.9.

Table 3.9

The Hyperparameters of SAC Approach for Motion Control Task

Hyperparameter	Setting Value
Episode number	500
Max step	400
Learning rate of policy network (α)	0.001
Learning rate of Q-function network (β)	0.001
Discount factor (γ)	0.99
Tau (τ)	0.001
Memory capacity (K)	10000
Minibatch (N)	128

3.5 Object Transportation

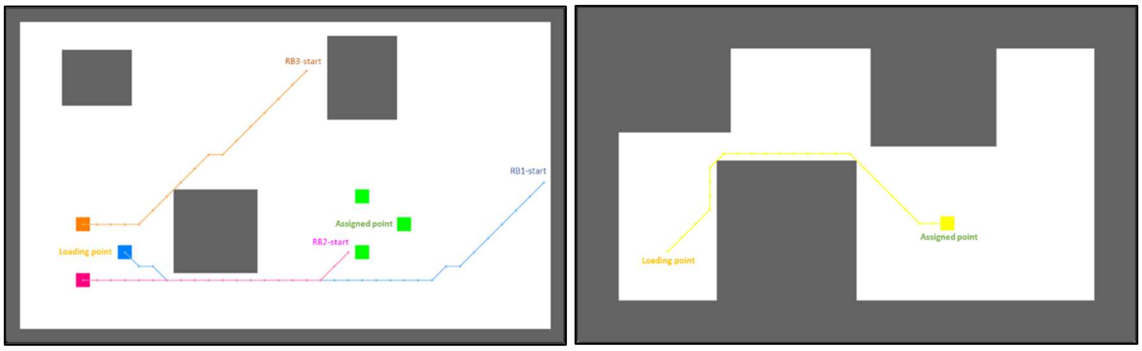
To accomplish the object transportation task, it is necessary to utilize the RL models both path planning and motion control which obtained from previous session. The models which have the best performance among all available options will be selected. As mentioned in session 3.1, the object transportation task consists of four sub-processes, finding paths for all robots to loading positions, controlling robots individually moving to target positions, finding path for group of robots from loading position to delivery position, and controlling group of robots moving in a specific formation with load carriage to target position.

Implementation in Simulation

This thesis evaluates the result of the RL models both in simulation and practice. However, it is impossible to load an object in simulation, so the last sub-process in simulation is replaced by displaying a group of robots moving in a specific formation instead. The program which evaluates the RL models is create using python. The RL model handling with path planning takes responsibility to find individual paths for all robots from arbitrary positions to specified positions at loading point. And it also finds path for a group of robots from loading position to delivery position (see Figure 3.19). For the motion control task, the RL agents are trained separately for two different purposes. The first RL model is trained to deal with path following task and another RL model is trained to handle with point following task.

Figure 3.*19*

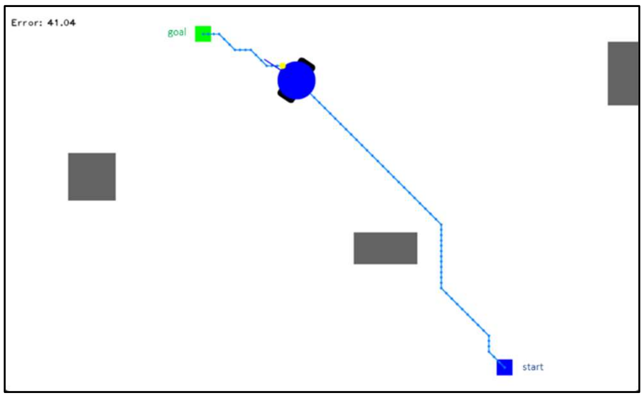
Path Planning for Individual Robot and Group of Robots.



In path following task, the agent is trained to control the robot's motion in order to follow the planned path from start position to goal position (see Figure 3.20). Several waypoints are created within path interval range. The robot's position error is computed using the current referred waypoint and the current position of the robot for estimating the robot's speeds in form of linear velocity and angular velocity. The robot's current position will be updated in real time to see the movement. When the distance error between robot and referred waypoint is less than 50 pixels, it shifts to refer the next waypoint in sequence. The episode will be finished when the agent can control robot to reach the goal position properly with error to the final position less than 2 pixels. Or the distance error goes beyond the acceptable error at 170 pixels, the episode will be forced to terminate.

Figure 3.20

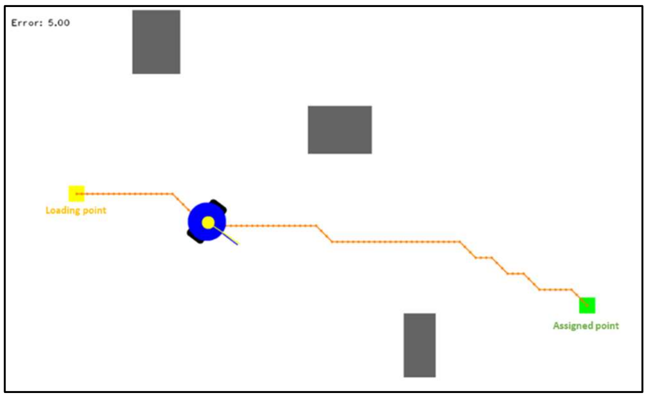
Motion Control Training Process for Path Following Task



Point following task is used to control robot to move in a specific formation when apply to multi-robot system. Considering when we would like the robots move in a specific formation, firstly, we set a formation with specified position for each robot. The center of group of robots is considered as the representative of this group. This thesis applied leader-followers concept to control multiple robots moving in specific formation. The virtual robot is created at the center of a formation as a leader and all robots are the followers. The reference points for all robots are created and updated in every step respected to the current position of this virtual robot which moving follow the path via PID control. The agent is trained by using the distance error between robot's current position and the moving reference point as an observation to approximate the action in linear velocity and angular velocity (see Figure 3.21). The agent will be obtained high reward when it can control the robot to move with less error as possible. The episode will end when the virtual robot reaches goal' position and the error between agent and last reference point is less than 2 pixels or the error is larger than an acceptable error at 120 pixels.

Figure 3.21

Motion Control Training Process for Point Following Task

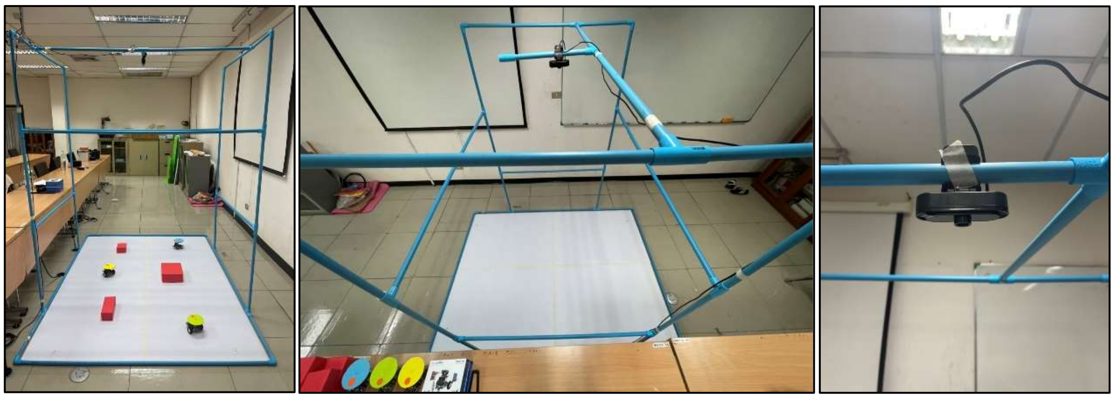


Implementation in Practice

When implement the RL models in practice, the map in real environment is set by placing the obstacles and mobile robots in a limited area. The map' dimension is 1200 millimeters in height and 2000 millimeters in width when considering in a perspective of webcam's view. However, the map can be changed by placing the obstacles and the robots in any available positions at initial state. The map was captured by single camera installed at middle top location of the map as see in figure 3.22.

Figure 3.22

Real Environment for Experiment.



The image captured by this webcam has a distortion effect, so it is necessary to undistort the image with distortion matrix obtained from camera calibration process (see Figure

3.23). Then the undistorted image is transformed using Homography matrix in order to have the perfect perspective of top view with specific area. The obstacles and robots are detected by using HSV range technique. the mask for each object is created using the defined values of HSV range, then the contours and positions of all objects are identified (see Figure 3.24).

Figure 3.23

Distorted Image and Undistorted Image

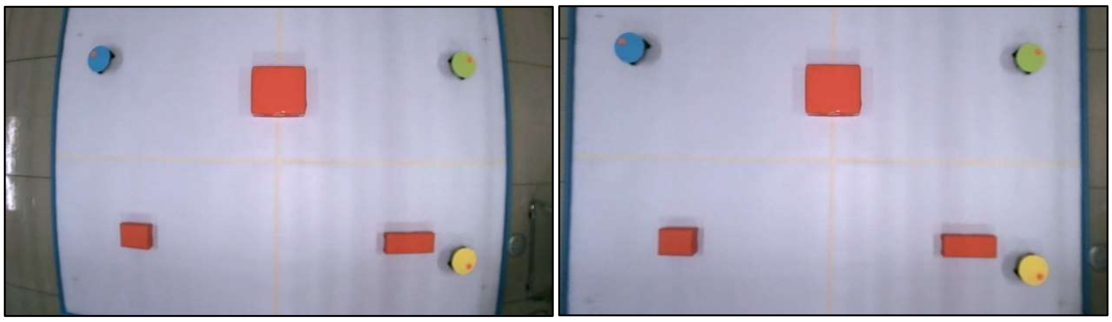
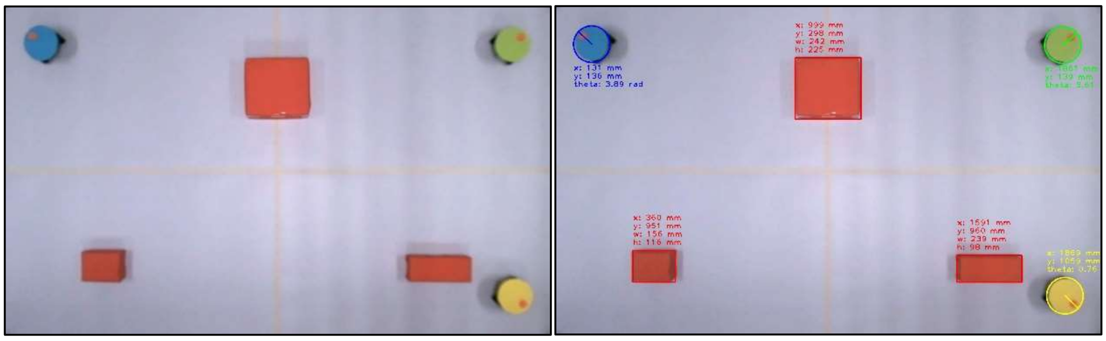


Figure 3.24

Image with Homography Transformation, and Identification of Position.



The processed image with map’s information is transformed into an observation for path planning agent in order to create the optimal path for each mobile robot. And the distance errors detected via camera are used to control the robots’ motion by the motion control agents. The results of performance comparison of the agents both for path planning and motion control task are shown in the evaluation chapter later.

CHAPTER 4
SIMULATION AND EXPERIMENT RESULTS

The object transportation of multiple robots is the consequence of assemblage of basic sub-processes of the robots. The details of cooperation among multiple robots in object transportation task are listed below:

1. Multiple robots are placed at any initial position inside the working area.
2. Reinforcement learning model finds the optimal paths with collision-free for all mobile robots individually from initial positions to loading points.
3. Reinforcement learning model controls each robot to follow its path to goal position.
4. The object is transferred to the robots.
5. Reinforcement learning model find the optimal path with collision-free for group of mobile robots from loading point to delivery point.
6. Reinforcement learning model controls all robots follow the path with fixed group shape to transfer the object to assign position.

To accomplish the object transportation, there are three necessary reinforcement learning models will be selected to handle to all subtasks, path planning, path following task, and point following task. In each subtask, several reinforcement learning model with different solutions will be evaluated in order to an appropriate reinforcement learning model.

4.1 Evaluate RL Model for Path Planning

In path planning, the performance of reinforcement learning model is evaluated by considering the capability of the model which can find the optimal path for mobile robot in any random map. Therefore, the RL model is evaluated by the episode reward that obtained in the environment of path planning. If the agent can find the shorter path from initial position to goal position that mean the higher reward, it could get. Moreover, the robustness of the model is measured by evaluating the RL model with several episodes to see how often it has fallen in a local optimum where the agent can't find path to the goal position. There are three steps were created in order to find the reinforcement learning model for path planning as following:

1. Evaluate the convolutional layers which suitable for the observation of this path planning.
2. Evaluate the reinforcement learning algorithm which has high performance in path planning.
3. Explore the effect of semi-supervised method by A* algorithm to the reinforcement learning model to find the optimal path.

Evaluate the Convolutional Layers

Three convolutional layers were selected as mentioned in previous chapter, the ordinary convolutional layers, Resnet18, and Resnet50. Those convolution networks were applied to deep Q-learning in order to solve path planning problem. During training process, the parameters of RL model will be saved when the agent obtained a new high score of average reward of last 100 episodes.

Figure 4.1 shows a moving average reward of DQN with ordinary convolutional layer in training process. The model obtained the highest average reward at episode 5766 with -11.35 scores. The model was ended training process at episode 8250 with 85 hours for training time, because this model had fallen into local optimum where agent always hit the obstacles and couldn’t find an appropriate path to goal position. This situation led the model to has a poor-quality Q-function and can’t further find path to the goal.

Figure 4.1

Training Result of DQN with Ordinary Convolutional Layer for Path Planning

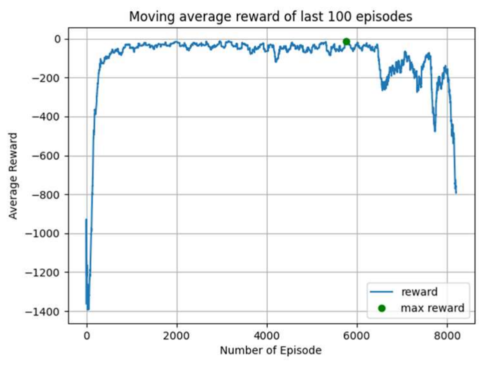


Figure 4.2 shows a result of DQN with ordinary convolutional layer in evaluating mode. the mean reward is -47.00. the maximum reward is 0.37. And the minimum reward is -600.00. this model can find the path for any simple maps. However, paths that were created by this model still far to consider as an optimal path. And it can't find path to goal position when encounter any complex maps.

Figure 4.2

Testing Result of DQN with Ordinary Convolutional Layer for Path Planning

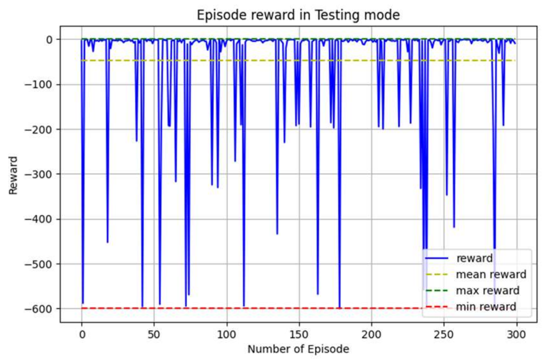


Figure 4.3 shows a moving average reward of DQN with Resnet18 in training process. The model obtained the highest average reward at episode 8145 with -8.50 scores. The training process took 84 hours 11 minutes and 14 seconds to complete 10,000 episodes.

Figure 4.3

Training Result of DQN with Resnet18 for Path Planning

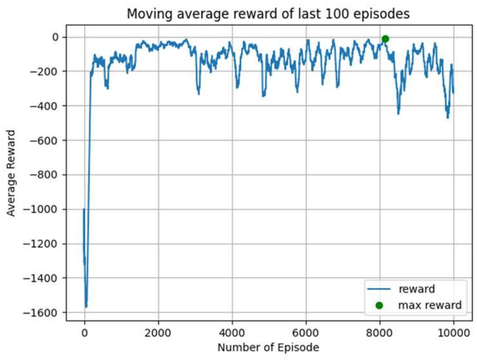


Figure 4.4 shows a result of DQN with Resnet18 in evaluating mode. the mean reward is -47.68. the maximum reward is 0.00. And the minimum reward is -600.00. this model can find the optimal path for some simple maps. However, many time, this model fell into local optimum near the goal position and couldn't find completed path.

Figure 4.4

Testing Result of DQN with Resnet18 for Path Planning

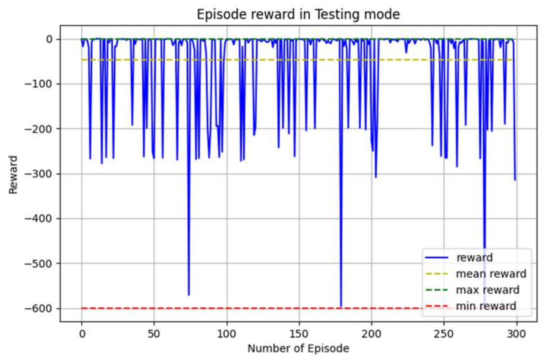


Figure 4.5 shows a moving average reward of DQN with Resnet50 in training process. The model obtained the highest average reward at episode 8093 with -5.65 scores. The training process took 59 hours 16 minutes and 57 seconds to complete 10,000 episodes. The reason for the shorter training time when compared to the DQN model with Resnet18 because this model was applied with smaller minibatch size than the DQN with Resnet18.

Figure 4.5

Training Result of DQN with Resnet50 for Path Planning

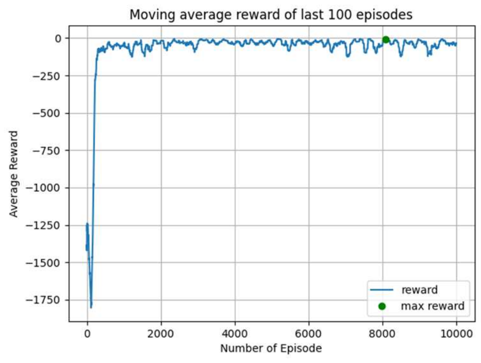
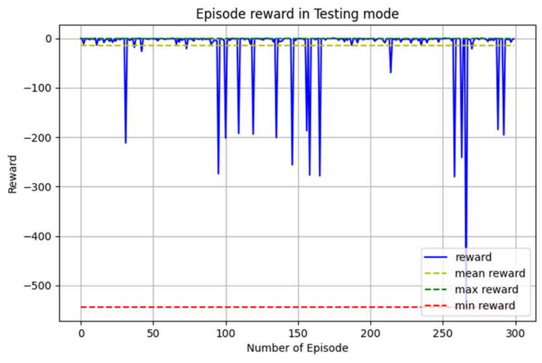


Figure 4.6 shows a result of DQN with Resnet50 in evaluating mode. the mean reward is -15.09. the maximum reward is 0.00. And the minimum reward is -544.36. this model can find path to the goal closely to the optimal path for both simple maps and complex map. However, it still had some episodes that the model can’t find an appropriate path to the goal.

Figure 4.6

Testing Result of DQN with Resnet50 for Path Planning



The performance comparison of DQN model with selected convolutional layers is shown in table 4.1. The result shows that deep Q-learning with Resnet50 is outperform both in training process and evaluating process when compared to the other choices. Even if the minibatch size setting in training process is less than the other but Resnet50 still has the best result. It can be concluded that Resnet50 can create better feature map which is easier to further estimate Q-values than other convolutional layers.

Table 4.1

Performance of DQN with Convolutional Layers for Path Planning

MODEL	TRAIN			TEST		
	BEST REWARD	EP	TRAIN TIME	MEAN REWARD	MAX REWARD	MIN REWARD
Simple Conv	-11.35	5766	Too long time	-47.00	-0.37	-600.00
Resnet18	-8.50	8145	303073.72	-47.68	0.00	-600.00
Resnet50	-5.65	8093	213416.94	-15.09	0.00	-544.36

Evaluate the RL Algorithms

Three reinforcement learning algorithms were selected as mentioned in previous chapter, deep Q-learning, double deep Q-learning, and proximal policy optimization. Those reinforcement learning model apply resnet50 as convolutional layer in order to solve path planning problem. During training process, the parameters of RL model will be saved when the agent obtained a new high score of average reward of last 100 episodes.

Figure 4.7 shows a moving average reward of DDQN with Resnet50 in training process. The model obtained the highest average reward at episode 9404 with -5.06 scores. The training process took 30 hours 44 minutes and 48 seconds to complete 10,000 episodes. The reason for the shorter training time when compared to the DQN model because this model was applied with smaller minibatch size than the DQN.

Figure 4.7

Training Result of DDQN with Resnet50 for Path Planning

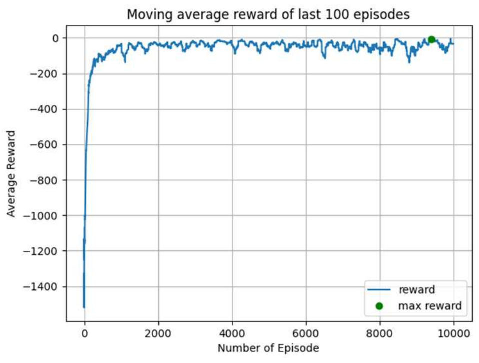


Figure 4.8 shows a result of DDQN with Resnet50 in evaluating mode. the mean reward is -10.34. the maximum reward is 0.00. And the minimum reward is -600.00. this model can find the optimal path for both simple maps and complex map. However, it still had just few episodes that model can't find an appropriate path to the goal. Nevertheless, the performance of DDQN algorithm is better than DQN algorithm if compared from the results.

Figure 4.8

Testing Result of DQN with Resnet50 for Path Planning

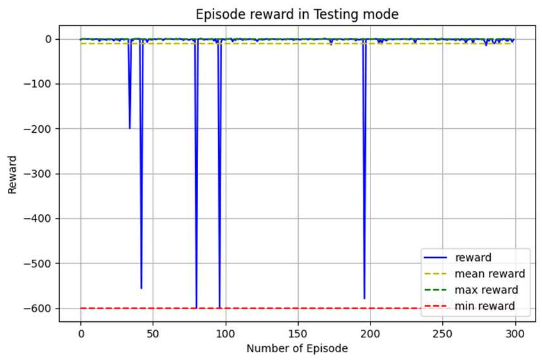


Figure 4.8 shows a moving average reward of PPO with Resnet50 in training process. The model obtained the highest average reward at episode 1190 with -931.45 scores. The training process was stopped at episode 1200 and took almost 65 hours. PPO with Resnet50 showed no signal to improve the performance since episode 200. the model can't converge and can't find path to the goal position as well.

Figure 4.9

Training Result of PPO with Resnet50 for Path Planning

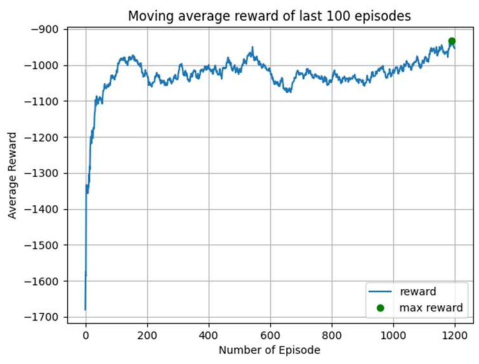
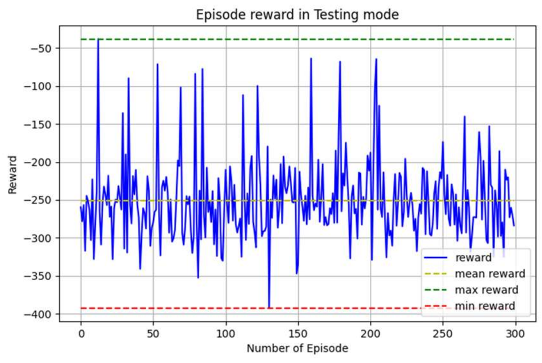


Figure 4.10 shows a result of PPO with Resnet50 in evaluating mode. the mean reward is -250.67. the maximum reward is 38.08. And the minimum reward is 392.23. Unlike DQN and DDQN, PPO model couldn't find path to goal position properly. Due to the model couldn't converge in training process. Due to PPO utilize the trajectory inside memory replay in learning process, but the memory capacity is very small compared to the maximum step per episode due to the computing capacity of the machine. It means that the model didn't have enough experience in exploration. And this led the model to have very low-quality Q-function.

Figure 4.10

Testing Result of PPO with Resnet50 for Path Planning



The performance comparison of reinforcement learning algorithms is shown in table 4.2. The result shows that double deep Q-learning with Resnet50 is outperform both in training process and evaluating process when compared to the other choices. Even if the minibatch size setting in training process of DDQN is less than DQN but it still has the best result. It can be concluded that DDQN has appropriate algorithm to estimate Q-values in path planning.

Table 4.2

Performance of RL Models for Path Planning

MODEL	**TRAIN**			**TEST**		
	BEST REWARD	**EP**	**TRAIN TIME**	**MEAN REWARD**	**MAX REWARD**	**MIN REWARD**
DQN	-5.65	8093	213416.94	-15.09	0.00	-544.36
DDQN	-5.06	9404	110687.63	-10.34	0.00	-600
PPO	-931.45	1190	Too long time	-250.67	-38.08	-392.23

Explore the Effect of Semi-Supervised by A* Algorithm

Normally, reinforcement learning model learns how to achieve the task and obtain the maximum total reward by exploration and exploitation. However, if we need the model to have a faster convergence rate, supervised method is a one choice that provide the optimum action directly to the environment and the trajectories will be collected in the memory for learning step later. This supervised method can reduce time consumption in exploration process in order to find the optimal action. In semi-supervised method, path in each episode were found by RL model and A* algorithm. The trajectories of both solutions are collected in memory. At the beginning, the trajectories collected from RL model aimed for exploration. And the trajectories collected from A* aimed for shortcutting to the optimal solution. However, the map always be changed during training process, so it is necessary to have enough experiences from exploration as well. In this experiment, the DDQN with semi-supervised by A* algorithm in various ratio will be tested. The parameters of RL model will be saved when the agent obtained a new high score of average reward of last 100 episodes.

Figure 4.11 shows a moving average reward of DDQN with 25% supervised by A* in training process. The model obtained the highest average reward at episode 9033 with -5.81 scores. The best score in training process didn't significantly difference to DDQN without supervised.

Figure 4.11

Training Result of DDQN with 25% Supervised by A for Path Planning*

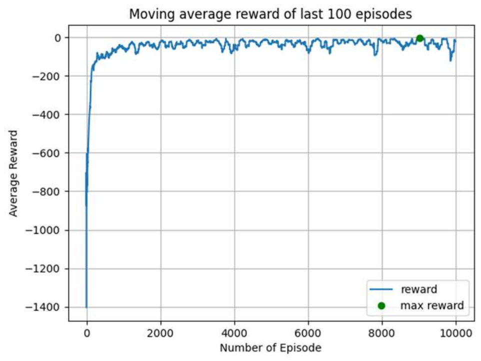


Figure 4.12 shows a result of DDQN with 25% supervised by A* in evaluating mode. the mean reward is -10.23. the maximum reward is 0.00. And the minimum reward is -591.91. this model can find the optimal path for both simple maps and complex map. However, it still had just few episodes that model can't find an appropriate path to the goal. The performance of DDQN with 25% supervised by A* is same as DDQN without supervised by A* but the training time is faster due to 25% of total episode were completed by A*.

Figure 4.12

Testing Result of DDQN with 25% Supervised by A for Path Planning*

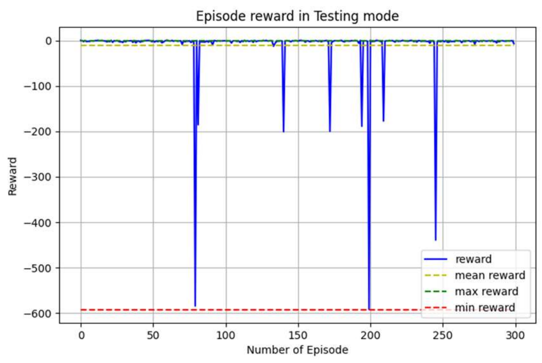


Figure 4.13 shows a moving average reward of DDQN with 50% supervised by A* in training process. The model obtained the highest average reward at episode 9501 with -2.47 scores. The best average reward of this mode is improved due to 50% of total paths were found by A* algorithm.

Figure 4.13

Training Result of DDQN with 50% Supervised by A for Path Planning*

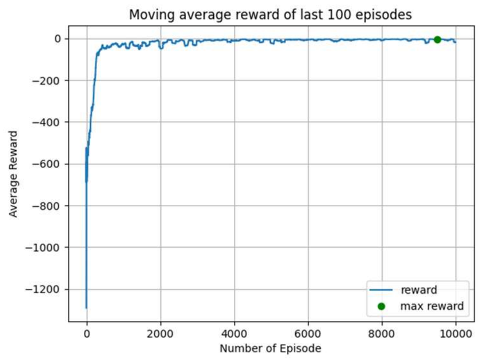


Figure 4.13 shows a result of DDQN with 50% supervised by A* in evaluating mode. the mean reward is -12.19. the maximum reward is 0.00. And the minimum reward is -600.00. this model can find the optimal path for both simple maps and complex map. However, it still had just few episodes that model can't find an appropriate path to the goal. The performance of DDQN with supervised 50% by A* is same as DDQN without supervised by A* but the training time is faster due to 50% of total episode were completed by A*.

Figure 4.14

Testing Result of DDQN with 50% Supervised by A for Path Planning*

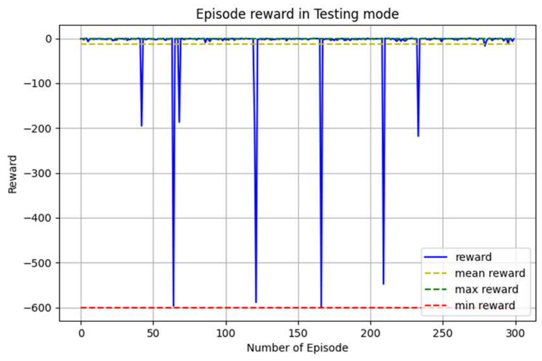


Figure 4.15 shows a moving average reward of DDQN with 75% supervised by A* in training process. The model obtained the highest average reward at episode 9483 with -2.10 scores. The best average reward of this mode is improved due 75% of total paths were found by A* algorithm.

Figure 4.15

Training Result of DDQN with 75% Supervised by A for Path Planning*

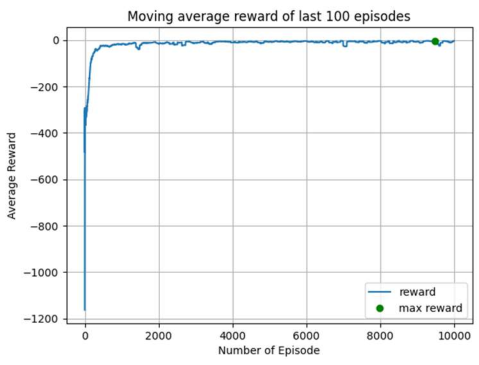
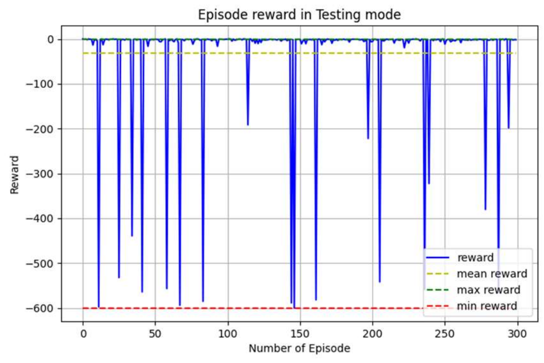


Figure 4.16 shows a result of DDQN with 75% supervised by A* in evaluating mode. the mean reward is -30.92. the maximum reward is 0.00. And the minimum reward is -591.91. this model can find the optimal path same as applying A* algorithm for simple maps. However, when the model encountered some complex map, many times it fell into local optimum and couldn’t find path to the goal position. Due to 75% of episodes in training process are accomplished by A* algorithm, thus this RL model lacks in experience from exploration in order to handle with any unseen environments.

Figure 4.16

Testing Result of DDQN with 75% Supervised by A for Path Planning*



The performance comparison of DDQN with semi-supervised by A* algorithm is shown in table 4.3. When considering the same number of episodes in training process, the result shows that double deep Q-learning with 25% supervised by A* has the highest score compared to the other choices. The performance of the RL model with different semi-supervisor ratio represented a trade-off between exploration and exploitation from A* algorithm. If the model has too much using A* algorithm in training process, it will lack in experience to handle with more complicated map. however, this problem can be solved by increasing the number of episodes in training process. Hence, the model could be improved to have an ability to find the optimal path like A* algorithm and can accomplish path planning for any random map.

Table 4.3

Performance of DDQN with Semi-Supervised by A for Path Planning*

MODEL	TRAIN			TEST		
	BEST REWARD	EP	TRAIN TIME	MEAN REWARD	MAX REWARD	MIN REWARD
0%	-5.06	9404	110687.63	-10.34	0.00	-600.00
25%	-5.81	9033	98791.01	-10.23	0.00	-591.91
50%	-2.47	9501	78166.37	-12.19	0.00	-600.00
75%	-2.10	9483	64841.65	-30.92	0.00	-600.00

4.2 Evaluate RL Model for Path Following Task

In path following task, the map is created in simulation randomly. A* algorithm is applied in order to find the optimal path. The objective of path following task is to control the differential wheeled mobile robot to follow the planned path until reach the goal. Three reinforcement learning algorithms in policy-based method are selected in order to test the performance in path following task. Those models consist of proximal policy optimization, deep deterministic policy gradient, and soft actor-critic. Reinforcement learning model which be trained in the environment without any disturbance will be used to perform in simulation. And reinforcement learning model which be trained in the environment with disturbance will be used to perform in practice.

Training in Environment without Disturbance

Figure 4.17 shows a moving average reward of PPO model in training process. The model obtained the highest average reward at episode 343 with 823.14 scores. The training process took 1 hour 49 minutes and 45 seconds to complete 500 episodes. PPO model can achieve path following task appropriately.

Figure 4.17

Training Result of PPO Model for Path Following Task (No Disturbance)

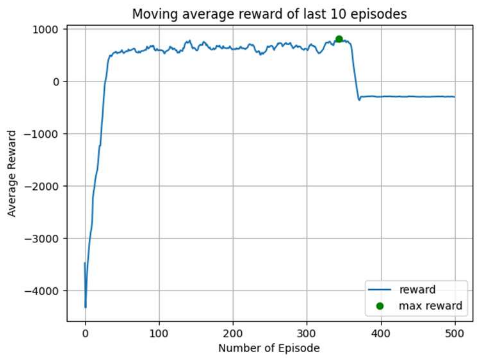


Figure 4.18 shows a result of PPO model in evaluating mode. the mean reward is 696.88. the maximum reward is 1088.60. And the minimum reward is 433.60. This model can control differential wheeled mobile robot follow path to the goal properly.

Figure 4.18

Testing Result of PPO Model for Path Following Task (No Disturbance)

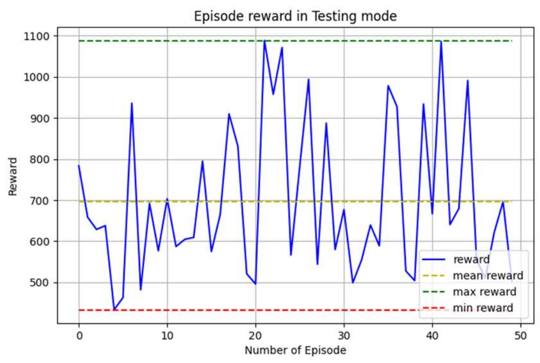


Figure 4.19 shows a moving average reward of DDPG model in training process. The model obtained the highest average reward at episode 372 with 770.89 scores. The training process took 2 hours 2 minutes and 59 seconds to complete 500 episodes. DDPG model can achieve path following task appropriately.

Figure 4.19

Training Result of DDPG Model for Path Following Task (No Disturbance)

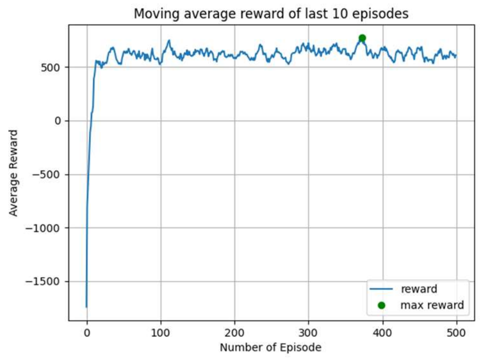


Figure 4.20 shows a result of DDPG model in evaluating mode. the mean reward is 525.16. the maximum reward is 789.34. And the minimum reward is 302.33. This model can control differential wheeled mobile robot follow path to the goal properly.

Figure 4.20

Testing Result of DDPG Model for Path Following Task (No Disturbance)

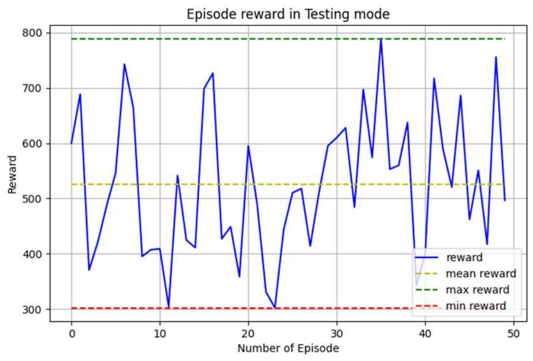


Figure 4.21 shows a moving average reward of SAC model in training process. The model obtained the highest average reward at episode 192 with 796.15 scores. The training process took 2 hours 29 minutes and 18 seconds to complete 500 episodes. SAC model can achieve path following task appropriately.

Figure 4.21

Training Result of DDPG Model for Path Following Task (No Disturbance)

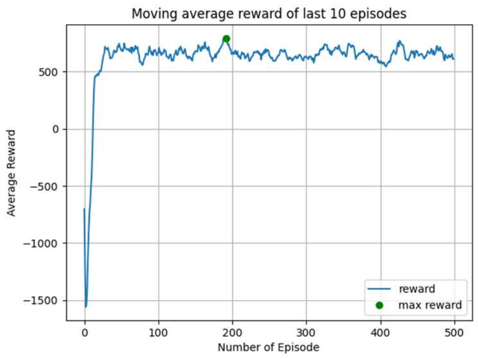
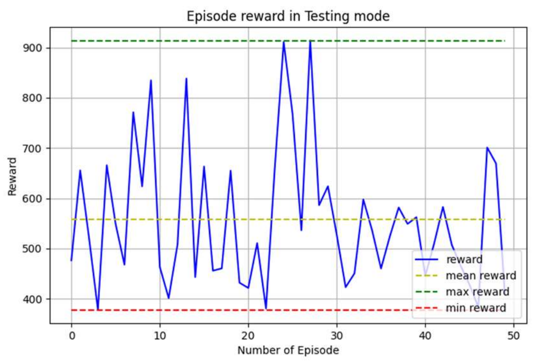


Figure 4.22 shows a result of SAC model in evaluating mode. the mean reward is 557.69. the maximum reward is 914.14. And the minimum reward is 377.90. This model can control differential wheeled mobile robot follow path to the goal properly.

Figure 4.22

Testing Result of SAC Model for Path Following Task (No Disturbance)



The performance comparison of reinforcement learning algorithms is shown in table 4.4. The result shows that proximal policy optimization is outperform both in training process and evaluating process when compared to the other choices. Therefore, PPO is selected to perform the path following task in simulation.

Table 4.4

Performance of RL Models for Path Following Task (No Disturbance)

MODEL	TRAIN			TEST		
	BEST REWARD	EP	TRAIN TIME	MEAN REWARD	MAX REWARD	MIN REWARD
PPO	823.14	343	6584.83	696.88	1088.60	433.60
DDPG	770.89	372	7379.68	525.16	789.34	302.33
SAC	796.15	192	8953.38	557.69	914.14	377.90

Training in Environment with Disturbance

Figure 4.23 shows a moving average reward of PPO model in training process. The model obtained the highest average reward at episode 355 with 786.92 scores. The training process took 2 hours 25 minutes and 45 seconds to complete 500 episodes. PPO model can achieve path following task in the environment with disturbance appropriately.

Figure 4.23

Training Result of PPO Model for Path Following Task (With Disturbance)

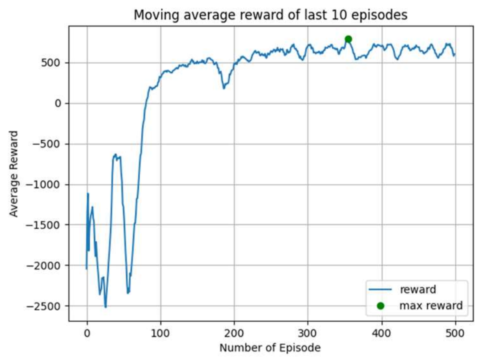


Figure 4.24 shows a result of PPO model in evaluating mode. the mean reward is 554.40. the maximum reward is 918.46. And the minimum reward is 373.27. This model can control differential wheeled mobile robot follow path to the goal in the environment with disturbance properly.

Figure 4.24

Testing Result of PPO Model for Path Following Task (With Disturbance)

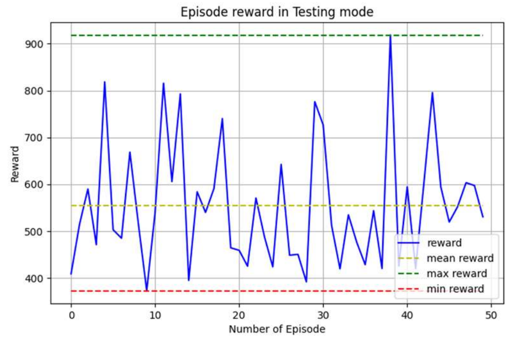


Figure 4.25 shows a moving average reward of DDPG model in training process. The model obtained the highest average reward at episode 331 with 705.04 scores. The training process took 2 hours 10 minutes and 28 seconds to complete 500 episodes. DDPG model can achieve path following task in the environment with disturbance appropriately.

Figure 4.25

Training Result of DDPG Model for Path Following Task (With Disturbance)

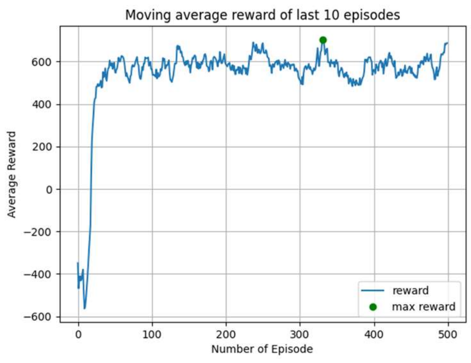


Figure 4.26 shows a result of DDPG model in evaluating mode. the mean reward is 507.78. the maximum reward is 830.97. And the minimum reward is 245.14. This model can control differential wheeled mobile robot follow path to the goal in the environment with disturbance properly.

Figure 4.26

Testing Result of DDPG Model for Path Following Task (With Disturbance)

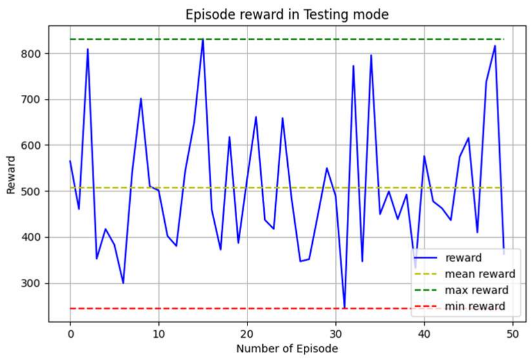


Figure 4.27 shows a moving average reward of SAC model in training process. The model obtained the highest average reward at episode 176 with 782.11 scores. The training process took 2 hours 29 minutes and 28 seconds to complete 500 episodes. SAC model can achieve path following task in the environment with disturbance appropriately.

Figure 4.27

Training Result of SAC Model for Path Following Task (With Disturbance)

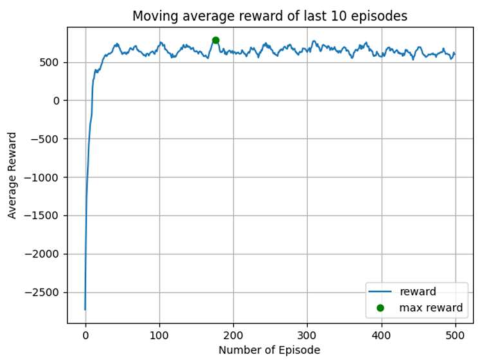
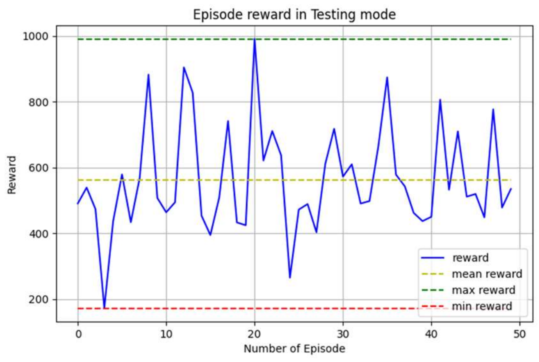


Figure 4.28 shows a result of SAC model in evaluating mode. the mean reward is 562.52. the maximum reward is 990.71. And the minimum reward is 172.29. This model can control differential wheeled mobile robot follow path to the goal in the environment with disturbance properly.

Figure 4.28

Testing Result of SAC Model for Path Following Task (With Disturbance)



The performance comparison of reinforcement learning algorithms is shown in table 4.5. There is no significant different in performance among selected algorithms. However, soft actor-critic has the highest reward in evaluation when compared to the other choices. Therefore, SAC is selected to perform the path following task in practice.

Table 4.5

Performance of RL Models for Path Following Task (With Disturbance)

MODEL	TRAIN			TEST		
	BEST REWARD	EP	TRAIN TIME	MEAN REWARD	MAX REWARD	MIN REWARD
PPO	786.92	355	8745.09	554.40	918.46	373.27
DDPG	705.04	331	7828.17	507.78	830.97	245.14
SAC	782.11	176	8968.46	562.52	990.71	172.29

4.3 Evaluate RL Model for Point Following Task

In point following task, Obstacles, loading position, and delivery position are randomly assigned when the map is created at initial. A* algorithm is applied in order to find the optimal path from loading position to delivery position. The reference point is created and be considered as a virtual mobile robot. This virtual robot is controlled by PID controller in order to follow the planned path until reach delivery position. The objective of point following task is to control the differential wheeled mobile robot to follow the moving reference point that be considered as a virtual robot, until reach delivery position. The performance of RL model is evaluated by considering the distance error between robot’s position and reference point. RL model must control the robot’s speed to move closer to this moving reference point as much as position. Three reinforcement learning algorithms in policy-based method are selected in order to test the performance in point following task. Those models consist of proximal policy optimization, deep deterministic policy gradient, and soft actor-critic. Reinforcement learning model which be trained in the environment without any disturbance will be used to perform in simulation. And reinforcement learning model which be trained in the environment with disturbance will be used to perform in practice.

Training in Environment without Disturbance

Figure 4.29 shows a moving average reward of PPO model in training process. The model obtained the highest average reward at episode 145 with 1121.62 scores. The training process took 1 hour 39 minutes and 43 seconds to complete 500 episodes. PPO model can achieve point following task appropriately.

Figure 4.29

Training Result of PPO Model for Point Following Task (No Disturbance)

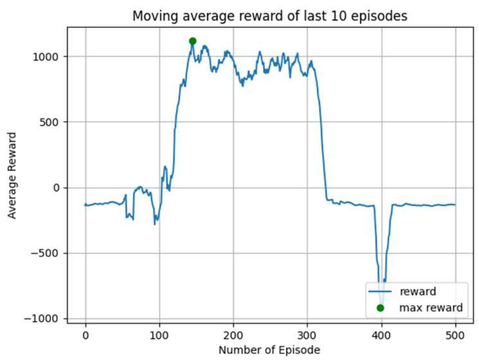


Figure 4.30 shows a result of PPO model in evaluating mode. The performance of RL model is measured by using the distance error. the mean error is 8.59. the maximum error is 9.80. And the minimum error is 7.27. This model can control differential wheeled mobile robot follow the reference point until reach the goal properly.

Figure 4.30

Testing Result of PPO Model for Point Following Task (No Disturbance)

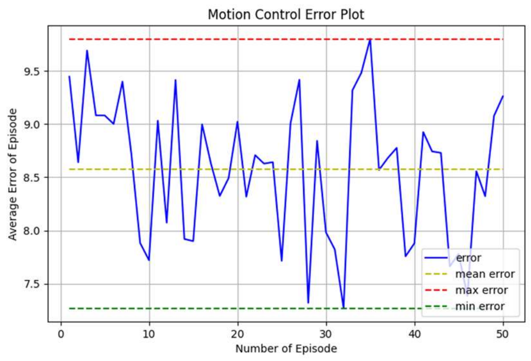


Figure 4.31 shows a moving average reward of DDPG model in training process. The model obtained the highest average reward at episode 272 with 1289.62 scores. The training process took 3 hour 3 minutes and 18 seconds to complete 500 episodes. DDPG model can achieve point following task appropriately.

Figure 4.31

Training Result of DDPG Model for Point Following Task (No Disturbance)

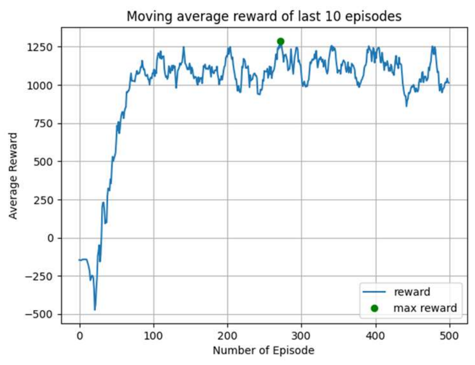


Figure 4.32 shows a result of DDPG model in evaluating mode. The performance of RL model is measured by using the distance error. the mean error is 3.44. the maximum error is 4.01. And the minimum error is 2.70. This model can control differential wheeled mobile robot follow the reference point until reach the goal properly.

Figure 4.32

Testing Result of DDPG Model for Point Following Task (No Disturbance)

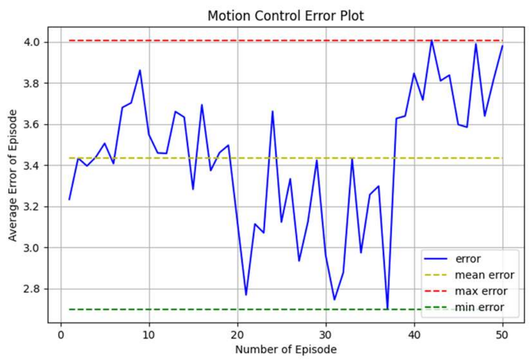


Figure 4.33 shows a moving average reward of SAC model in training process. The model obtained the highest average reward at episode 120 with 1086.43 scores. The training process took 3 hour 15 minutes and 21 seconds to complete 500 episodes. SAC model can achieve point following task appropriately.

Figure 4.33

Training Result of SAC Model for Point Following Task (No Disturbance)

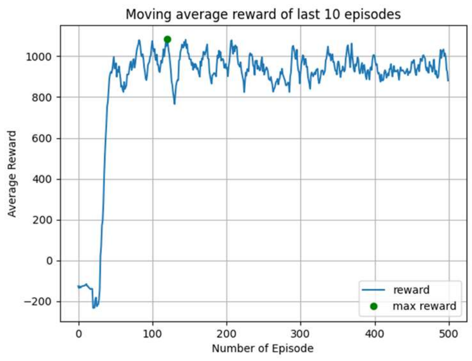
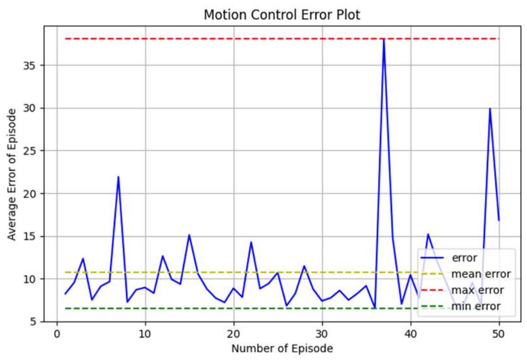


Figure 4.34 shows a result of SAC model in evaluating mode. The performance of RL model is measured by using the distance error. the mean error is 10.71. the maximum error is 38.04. And the minimum error is 6.55. This model can control differential wheeled mobile robot follow the reference point until reach the goal properly.

Figure 4.34

Testing Result of SAC Model for Point Following Task (No Disturbance)



The performance comparison of reinforcement learning algorithms is shown in table 4.6. The result shows that deep deterministic policy gradient is outperform both in training process and evaluating process when compared to the other choices. Therefore, DDPG is selected to perform the point following task in simulation.

Table 4.6

Performance of RL Models for Point Following Task (No Disturbance)

MODEL	TRAIN			TEST		
	BEST REWARD	EP	TRAIN TIME	MEAN ERROR	MAX ERROR	MIN ERROR
PPO	1121.62	145	5983.04	8.58	9.80	7.27
DDPG	1289.62	272	10997.53	3.44	4.01	2.70
SAC	1086.43	120	11721.35	10.71	38.04	6.55

Training in Environment with Disturbance

Figure 4.35 shows a moving average reward of PPO model in training process. The model obtained the highest average reward at episode 294 with 967.47 scores. The training process took 1 hour 42 minutes and 52 seconds to complete 500 episodes. PPO model can achieve point following task in the environment with disturbance appropriately.

Figure 4.35

Training Result of PPO Model for Point Following Task (With Disturbance)

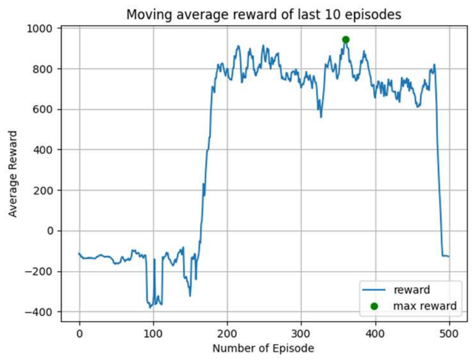


Figure 4.36 shows a result of PPO model in evaluating mode. The performance of RL model is measured by using the distance error. the mean error is 11.59. the maximum error is 16.72. And the minimum error is 7.88. This model can control differential wheeled mobile robot follow the reference point until reach the goal in the environment with disturbance properly.

Figure 4.36

Testing Result of PPO Model for Point Following Task (With Disturbance)

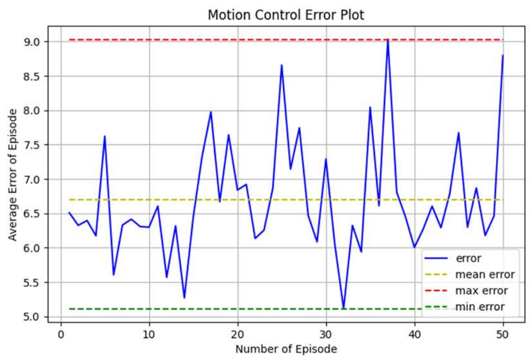


Figure 4.37 shows a moving average reward of DDPG model in training process. The model obtained the highest average reward at episode 129 with 796.75 scores. The training process took 2 hour 41 minutes and 18 seconds to complete 500 episodes. DDPG model can achieve point following task in the environment with disturbance. However, there are several times that this model controlled the mobile robot to rotate then follow the point by moving backward. And this is an undesirable situation.

Figure 4.37

Training Result of DDPG Model for Point Following Task (With Disturbance)

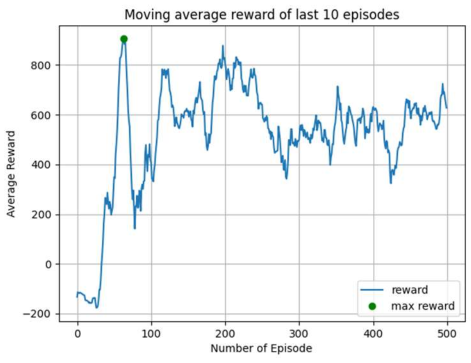


Figure 4.38 shows a result of DDPG model in evaluating mode. The performance of RL model is measured by using the distance error. the mean error is 8.48. the maximum error is 28.80. And the minimum error is 3.81. This model can control differential wheeled mobile robot follow the reference point until reach the goal in the environment with but several time the robot was controller to move backward in order to follow the reference point instead.

Figure 4.38

Testing Result of DDPG Model for Point Following Task (With Disturbance)

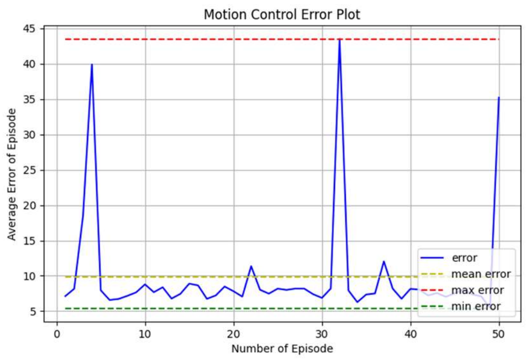


Figure 4.39 shows a moving average reward of SAC model in training process. The model obtained the highest average reward at episode 448 with 970.62 scores. The training process took 3 hour 23 minutes and 27 seconds to complete 500 episodes. SAC model can achieve point following task in the environment with disturbance appropriately.

Figure 4.39

Training Result of SAC Model for Point Following Task (With Disturbance)

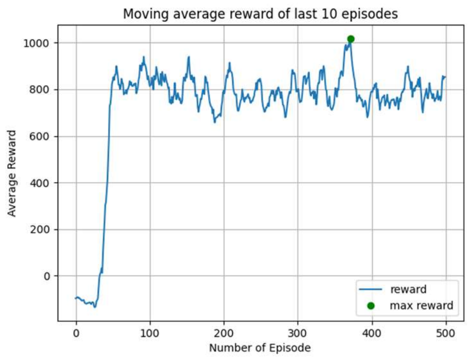
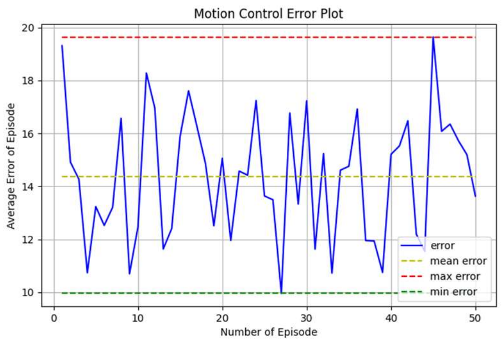


Figure 4.40 shows a result of SAC model in evaluating mode. The performance of RL model is measured by using the distance error. the mean error is 13.85. the maximum error is 23.10. And the minimum error is 8.5. This model can control differential wheeled mobile robot follow the reference point until reach the goal in the environment with disturbance properly.

Figure 4.40

Testing Result of SAC Model for Point Following Task (With Disturbance)



The performance comparison of reinforcement learning algorithms is shown in table 4.7. The result shows that even if deep deterministic policy gradient has the lowest mean error in evaluation but its maximum error is too high compare the other. It is caused by rotation during movement in order to follow the reference point with backward direction. In this case, proximal policy optimization is selected to perform the point following task in practice.

Table 4.7

Performance of RL Models for Point Following Task (With Disturbance)

MODEL	TRAIN			TEST		
	BEST REWARD	EP	TRAIN TIME	MEAN ERROR	MAX ERROR	MIN ERROR
PPO	967.47	294	6172.42	11.59	16.72	7.88
DDPG	796.75	129	9678.17	8.48	28.80	3.81
SAC	970.62	448	12206.50	13.85	23.10	8.75

4.4 Implement RL Models for Object Transportation

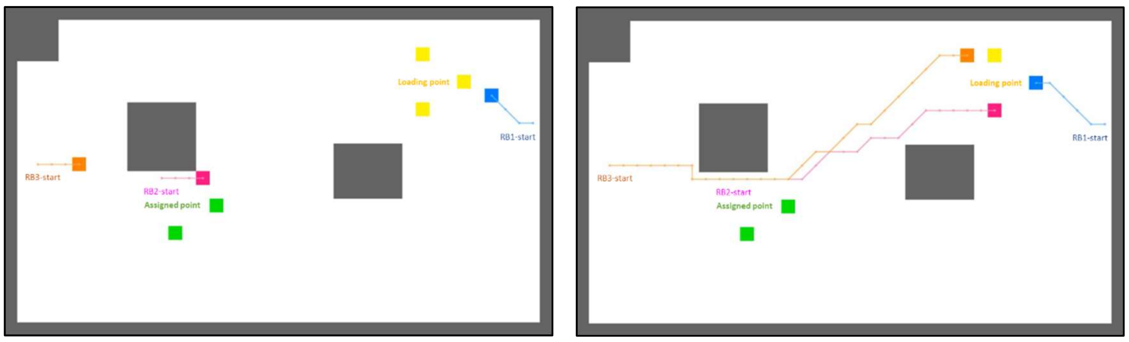
After completely evaluate the performances of reinforcement learning algorithms for all tasks. Double deep Q-learning with Resnet10 which 25% of training episode is supervised by A* is selected to achieve path planning for individual robots and group of robots. For path following task, proximal policy optimization is select to perform in simulation and soft actor-critic is selected to execute in practice. For point following task, deep deterministic policy gradient is selected to perform in simulation and proximal policy optimization is selected to execute in practice. All selected RL models are assembled in order to accomplish the object transportation task both in simulation and practice.

Simulation Result of Multiple Robots

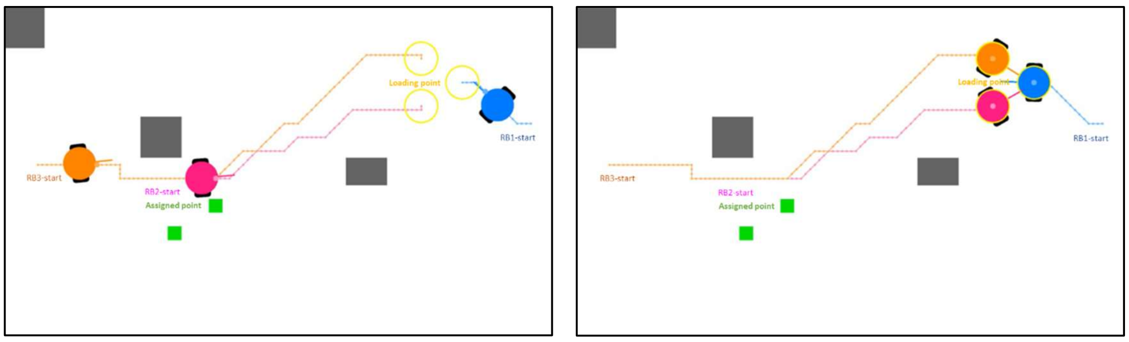
The RL models are tested firstly in simulation in order to see the performance in the environment without any disturbance. Figure 4.41 shows sample of simulation result when implement the RL models among multiple robots in object transportation.

Figure 4.41

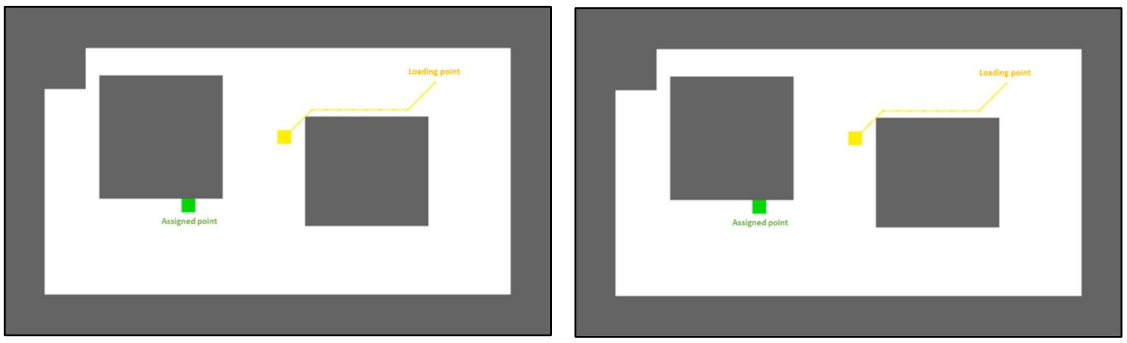
Simulation Result of Multiple Robots for Object Transportation



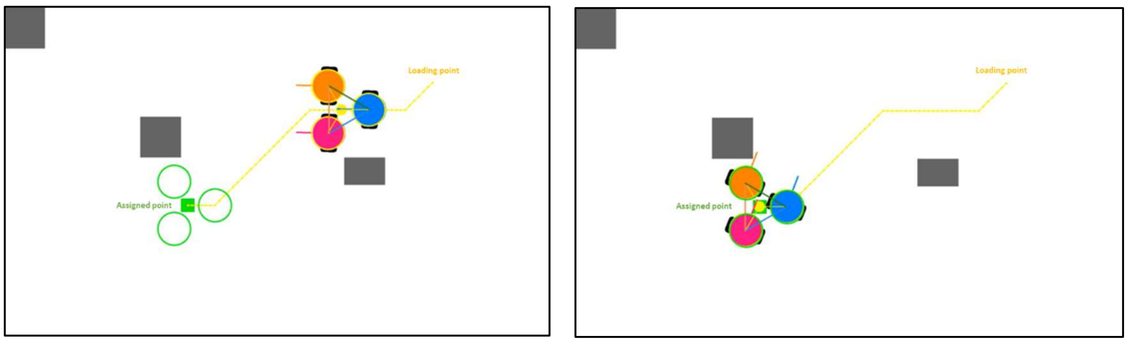
Path Planning for Individual Robots



Path Following for Individual Robots



Path Planning for Group of Robots



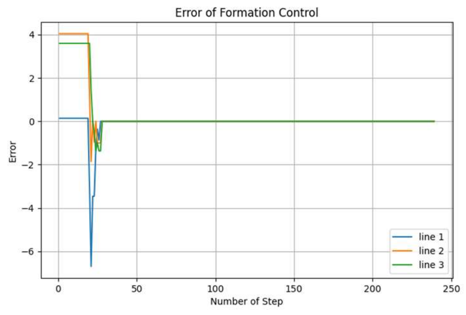
Formation Control for Group of Robots

Firstly, three mobile robots stand at the initial position at arbitrary point inside map's area. The loading point is assigned randomly. The goal's position of each robot is generated respected to the robot's formation. DDQN model finds the optimal paths for all robots to their goal's positions. The waypoints are created according to each robot's path. PPO model for path following task controls the robots to follow their paths using current waypoint to calculate errors then estimates a linear velocity and angular velocity for a current step. After all robots reach their loading positions. DDQN model finds the optimal path for group of robots from loading point to delivery point. The virtual robot is created at the middle of the formation to represent the group of mobile robots. This virtual robot is controlled to follow the path to delivery point by PID controller. In every time step, the reference point of each robot is computed refer to the current position of the virtual robot. Then DDPG of point following task controls all robots to follow their reference points in order to create movement of group of robots with fixed group shape.

Figure 4.42 shows the error of formation control in each time step. If the robots move with specific formation, the distance between each robot should be maintained. Hence, the distance between robot no.1 and robot no.2 is considered as line1. The distance between robot no.2 and robot no.3 is considered as line2. The distance between robot no.3 and robot no.1 is considered as line3. This graph shows the errors between real distance and ideal distance occurred in line 1-3. When there is no disturbance in the environment, RL models have an excellent performance.

Figure 4.42

Formation Errors of Multiple Robots in Simulation

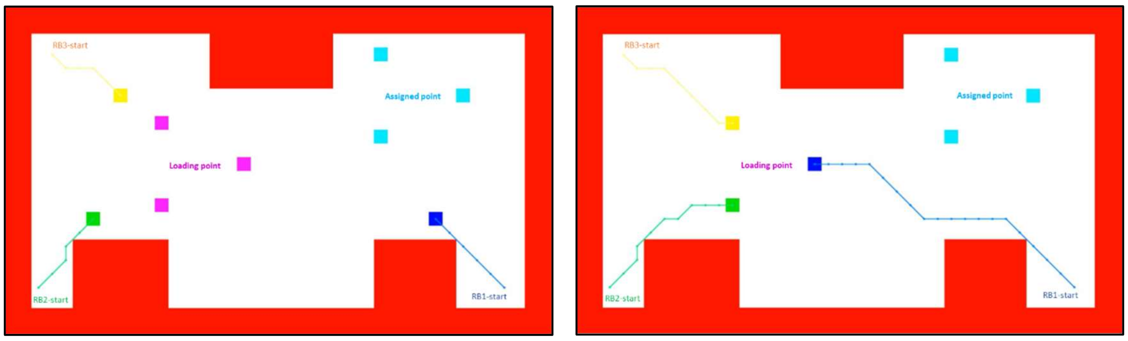


Experiment Result of Multiple Robots without Object

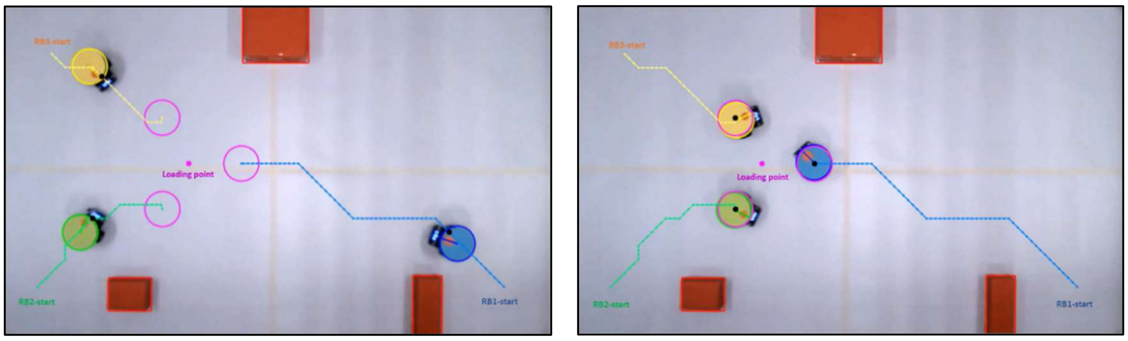
The RL models are tested in practice in order to see the performance in the environment with many disturbances and uncontrollable factors. For examples, a speed of mobile robot doesn't match exactly to the command speed because this thesis used an output control system which is different to a feedback loop control system. Moreover, there is a delay time when broadcasting the command from the server to the robots. However, the main problem is the normal speed of the selected robot of this thesis is too fast compared to the available environment, thus it is necessary to apply a very low speed of the robot and set many conditions in order to implement the RL models in practice. Figure 4.43 shows sample of experiment result when implement the RL models among multiple robots in formation control task.

Figure 4.43

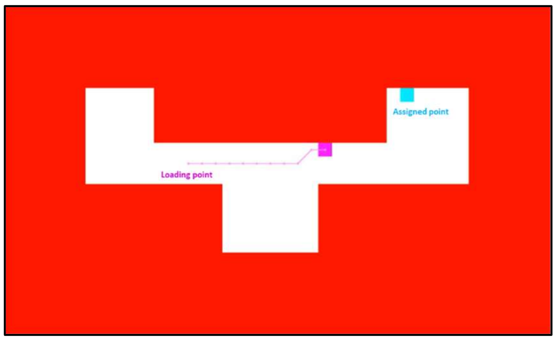
Experiment Result of Multiple Robots for Formation Control Task

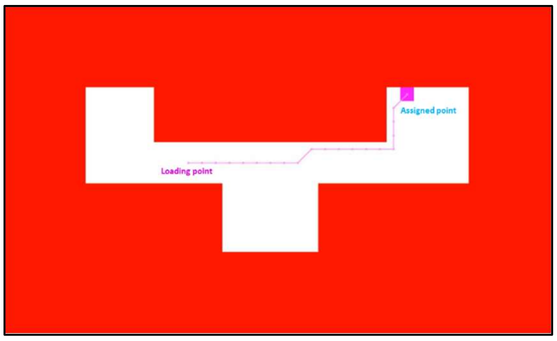


Path Planning for Individual Robots

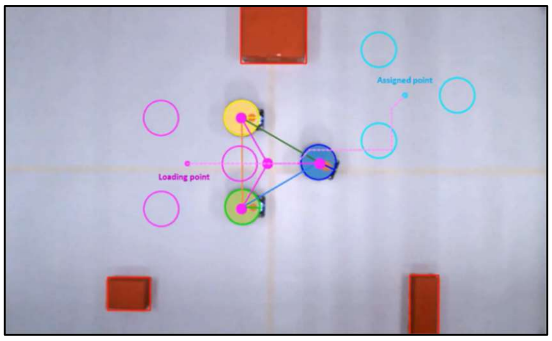


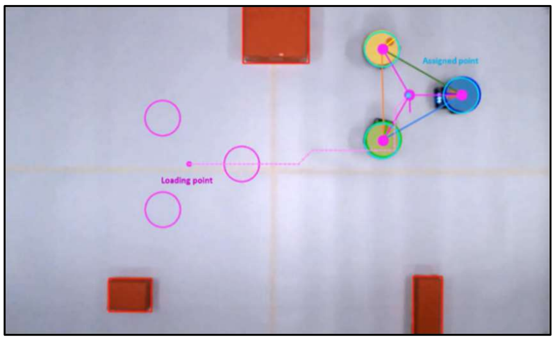
Path Following for Individual Robots





Path Planning for Group of Robots



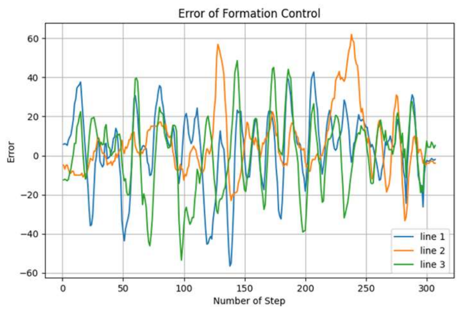


Formation Control for Group of Robots

The process of implementation in experiment is the same as showing in simulation result. However, several conditions are set in order to implement the RL models appropriately in practice. For example, the formation of group of robots is expanded in order to prevent an incident that robot crashes to each other during moving in a formation. Figure 4.44 shows the error of formation control in each time step when deploys the RL models in experiment. The formation error is ranged between -60 to 60 mm. it is acceptable for an output control system.

Figure 4.44

Formation Errors of Multiple Robots in Experiment (Excluded the Object)

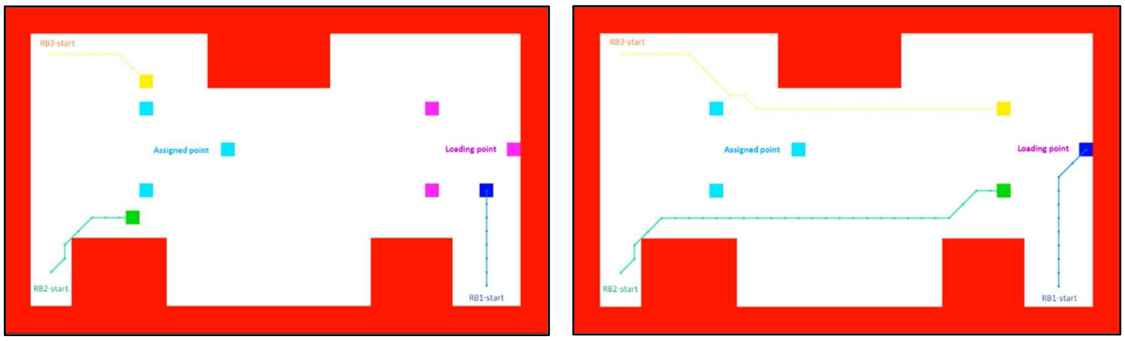


Experiment Result of Multiple Robots with the Object

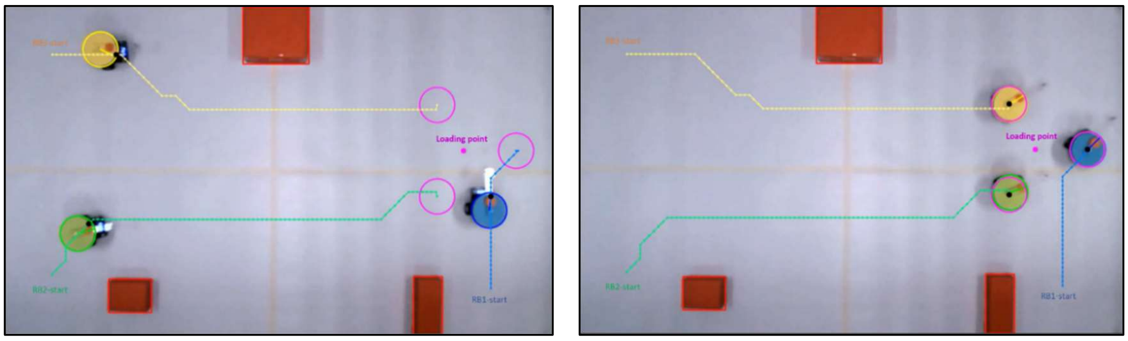
The RL models are tested in practice with the object in order to see the performance. Figure 4.45 shows sample of experiment result when implement the RL models among multiple robots in object transportation.

Figure 4.45

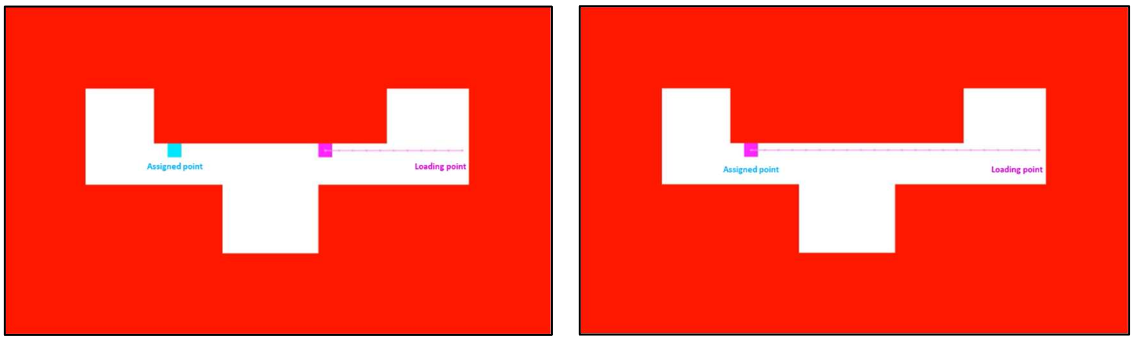
Experiment Result of Multiple Robots for Object Transportation



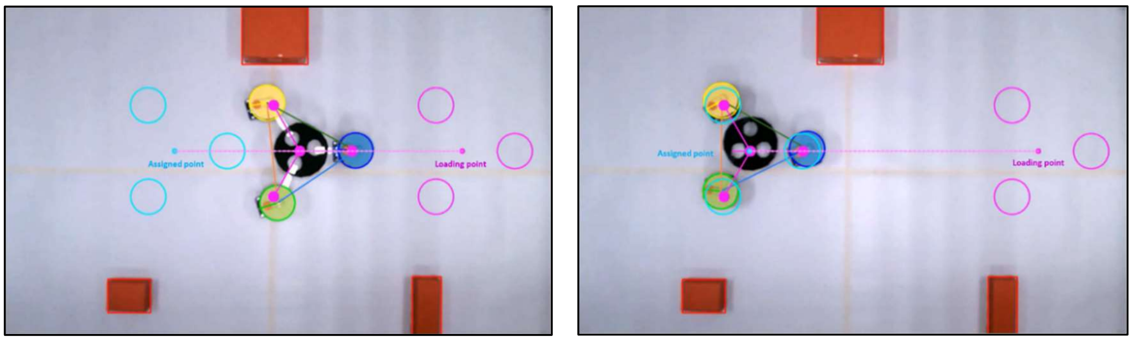
Path Planning for Individual Robots



Path Following for Individual Robots



Path Planning for Group of Robots

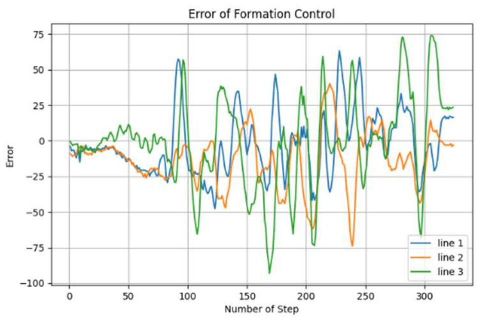


Formation Control for Group of Robots

The result of implementing RL models in experiment with the object shows the same performance compare when it has no object. Nevertheless, the formation error in the object transportation is ranged between -80 to 75 mm. it is larger when compared to the result in a formation control task due to the load that exerts to the robots in different positions is effect to robots' motion see in figure 4.45.

Figure 4.46

Formation Errors of Multiple Robots in Experiment (Included the Object)



CHAPTER 5
CONCLUSION

This thesis proposed reinforcement learning model to find the optimal path for any random map. According to the experiment results, double deep Q-learning with Resnet50 has the best performance when compared to other selected reinforcement learning algorithms. Moreover, semi-supervised method by A* algorithm can increase the performance of reinforcement learning model in order to find the optimal path. Whereas it decreases an experience of exploration in memory replay buffer. Therefore, it is necessary to compensate a lacking in exploration by training the model with enough number of episodes. Then the capability of the model could be improved. In addition, this thesis also proposed another reinforcement learning models to control the motion of differential wheeled mobile robot both for path following task and point following task. The reinforcement learning model which handle with point following task is used to control a group of mobile robots to move in a specific formation. Due to there is no significant difference of performance among selected reinforcement learning algorithms in evaluation. Hence, the reinforcement learning models which have the highest score are selected to perform each task both in simulation and experiment. The integration of proposed reinforcement learning models are used to perform for object transportation in both simulation and experiment. The map's area is enveloped within 1.2 x 2.0 square meter. The obstacles are randomly placed within map's area. Robot's initial positions, loading position, and delivery position are assigned in arbitrary position. The environment's information in experiment was obtained using single camera which applied HSV range technique in real time process. Eventually, the proposed reinforcement learning models can accomplish the object transportation task appropriately both in simulation and in experiment.

The recommendations of this thesis can be listed as the following.

- The reinforcement learning model for path planning should be tested with more complicated environments.
- The reinforcement learning model with other convolutional layers should be explored further in order to improve the performance of the model.

- This thesis solved a path planning problem with static environment. However, it is interesting to do path planning in dynamic environments as well.
- Lastly, the reinforcement learning model which can simultaneously create a path planning and control a motion of group of mobile robots should be studied further.

REFERENCES

Abiyev, R., Ibrahim, D., & Erin, B. (2010). Navigation of mobile robots in the presence of obstacles. Advances in Engineering Software, 41(10–11), 1179–1186. https://doi.org/10.1016/j.advengsoft.2010.08.001

Desai, J. P., Ostrowski, J., & Kumar, V. (1998). Controlling formations of multiple mobile robots. Proceedings - IEEE International Conference on Robotics and Automation, 4, 2864–2869. https://doi.org/10.1109/ROBOT.1998.680621

Duchon, F., Babinec, A., Kajan, M., Beno, P., Florek, M., Fico, T., & Jurišica, L. (2014). Path planning with modified A star algorithm for a mobile robot. Procedia Engineering, 96, 59–69. https://doi.org/10.1016/j.proeng.2014.12.098

Feng, Z., Hu, G., Sun, Y., & Soon, J. (2020). An overview of collaborative robotic manipulation in multi-robot systems. Annual Reviews in Control, 49, 113–127. https://doi.org/10.1016/j.arcontrol.2020.02.002

Goh, K., & Tjahjono, B. (2006). A review of research in manufacturing. Informatics, 2006 IEEE, 00, 417–422. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4053424

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. 35th International Conference on Machine Learning, ICML 2018, 5, 2976–2989.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December, 770–778. https://doi.org/10.1109/CVPR.2016.90

Huang, Y. (2020). Deep Q-networks. Deep Reinforcement Learning: Fundamentals, Research and Applications, 135–160. https://doi.org/10.1007/978-981-15-4095-0_4

Janglová, D. (2004). Neural networks in mobile robot motion. In International Journal of Advanced Robotic Systems (Vol. 1, Issue 1, pp. 15–22). https://doi.org/10.5772/5615

Kala, R. (2014). Coordination in navigation of multiple mobile robots. Cybernetics and Systems, 45(1), 1–24. https://doi.org/10.1080/01969722.2014.862085

Kanayama, Y. (1990). A Stable Tracking Control Method for an Autonomous Mobile Robot. 384–389.

Khoshnevis, B., & Bekey, G. (1998). Centralized sensing and control of multiple mobile robots. Computers and Industrial Engineering, 35(3–4), 503–506. https://doi.org/10.1016/s0360-8352(98)00144-2

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings.

Mazyavkina, N., Sviridov, S., Ivanov, S., & Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. Computers and Operations Research, 134(August 2020), 105400. https://doi.org/10.1016/j.cor.2021.105400

Patle, B. K., Babu L, G., Pandey, A., Parhi, D. R. K., & Jagadeesh, A. (2019). A review: On path planning strategies for navigation of mobile robot. Defence Technology, 15(4), 582–606. https://doi.org/10.1016/j.dt.2019.04.011

Recker, T., Heinrich, M., & Raatz, A. (2020). A Comparison of Different Approaches for Formation Control of Nonholonomic Mobile Robots regarding Object Transport. Procedia CIRP, 96, 248–253. https://doi.org/10.1016/j.procir.2021.01.082

Richard Bellman. (1957). A Markovian decision process.

Sartoretti, G., Kerr, J., Shi, Y., Wagner, G., Satish Kumar, T. K., Koenig, S., & Choset, H. (2019). PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. IEEE Robotics and Automation Letters, 4(3), 2378–2385. https://doi.org/10.1109/LRA.2019.2903261

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. 1–12. http://arxiv.org/abs/1707.06347

Shim, J. H., & Cho, Y. I. (2015). A mobile robot Localization using external surveillance cameras at indoor. Procedia Computer Science, 56(1), 502–507. https://doi.org/10.1016/j.procs.2015.07.242

Wang, B., Liu, Z., Li, Q., & Prorok, A. (2020). Reinforcement Learning. 5(4), 6932–6939.

Wen, S., Wen, Z., Zhang, D., Zhang, H., & Wang, T. (2021). A multi-robot path-planning algorithm for autonomous navigation using meta-reinforcement learning based on transfer learning. Applied Soft Computing, 110, 107605. https://doi.org/10.1016/j.asoc.2021.107605

Xiao, Y., Hoffman, J., Xia, T., & Amato, C. (2020). Learning Multi-Robot Decentralized Macro-Action-Based Policies via a Centralized Q-Net. Proceedings - IEEE International Conference on Robotics and Automation, 10695–10701. https://doi.org/10.1109/ICRA40945.2020.9196684