INTELLIGENT MOTION PLANNING OF A MOBILE ROBOT BY USING CONVOLUTIONAL NEURAL NETWORK

SITI ASMAH BINTI ABDULLAH

RESEARCH REPORT SUBMITTED TO THE FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA, IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING
(MECHATRONICS)
2018

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: SITI ASMAH BINTI ABDULLAH

Registration/Matric No: KQF 160013

Name of Degree: MASTER OF ENGINEERING (MECHATRONICS)

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

INTELLIGENT MOTION PLANNING OF A MOBILE ROBOT BY USING CONVOLUTIONAL NEURAL NETWORK

Field of Study: Mechatronics Engineering

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every right in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be the owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work, I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name:

Designation:

ABSTRACT

Optimizing the cost in every aspect is important in the implementation of artificial intelligence (A.I.) without affecting the accuracy of the output. Especially when it related to wastage of energy towards environment treats and resources cost. Planning the path of the mobile robot by using Convolutional Neural Network (CNN) is important in optimizing the cost in terms of energy, time and human intervention force. The robot should be able to decide the correct and safe movement depending on its current position, path reference and obstacles' location. The mobile robot should be continuously predicting the correct movement until it reaches the targeted location by avoiding all the obstacles along its way. The intelligence of the mobile robot is imitating CNN where the network consists of Convolutional Layer, Normalization Layer, ReLU layer, Fully-connected Layer, Softmax Layer and the last layer is Classification Layer. The network needs to undergo training session before handling the mobile robot movement. It trains by using labelled data which comes by eight different folders represent the eight different movements; up, up-right, right, right-down, down-left, left, and left-up. By the path reference created by A* algorithm the robot is capable in optimizing its path to reach the designated destination. The mobile robot is being tested in three different environment maps which come with unique and different levels of difficulty. Every map contains of static obstacle arranged in the horizontal, vertical and diagonal manners. The robot should be able to avoid the obstacle during in its path, but if the collision happens, the robot should start from its initial position, and the CNN requires to re-train to avoid the same looping decision is made. Capturing the current position of the mobile robot is very important in determining the next best move to be taken. Windowing Grid is proposed to

solve this matter. Several algorithms are used to create the Windowing Grid in capturing the informative and relevant current position of the mobile robot. This information will be used by CNN to predict the best move as the process is repeated until the robot reaches the target position.

ABSTRAK

Mengoptimumkan kos dalam setiap aspek adalah penting dalam pelaksanaan kecerdasan buatan tanpa mempengaruhi ketepatan output. Terutama apabila ia berkaitan dengan pembaziran tenaga ke arah merawat persekitaran dan kos sumber. Merancang jalan robot mudah alih dengan menggunakan Rangkaian Neural Convolutional (CNN) adalah penting dalam mengoptimumkan kos dari segi tenaga, masa dan tenaga campur tangan manusia. Robot sepatutnya dapat menentukan pergerakan yang betul dan selamat bergantung pada kedudukan semasa, rujukan laluan dan lokasi rintangan. Robot mudah alih perlu meramal pergerakan yang betul sehingga ia mencapai lokasi yang disasarkan dengan mengelakkan semua halangan di sepanjang jalannya. Perisikan robot mudah alih meniru Rangkaian Neural Convolutional di mana rangkaian tersebut terdiri daripada Lapisan Convolutional, Lapisan Normalisasi, Lapisan ReLU, Lapisan Sepenuhnya Terhubung, Lapisan Softmax dan Lapisan Klasifikasi yang terakhir. Rangkaian ini perlu menjalani sesi latihan sebelum mengendalikan pergerakan robot bergerak. Ia melatih dengan menggunakan data berlabel yang datang oleh lapan folder yang berbeza mewakili lapan pergerakan yang berbeza; atas, atas-kanan, kanan, kanan-bawah, bawah-kiri, kiri, kiri-atas. Dengan rujukan laluan dibuat oleh algoritma A * robot mampu mengoptimumkan laluannya untuk mencapai destinasi yang ditetapkan. Robot mudah alih sedang diuji dalam tiga peta persekitaran yang berbeza yang datang dengan tahap kesukaran yang unik dan berbeza. Setiap peta mengandungi halangan statik yang diatur dalam gaya mendatar, menegak dan pepenjuru. Robot sepatutnya dapat mengelak rintangan semasa dalam laluannya, tetapi jika perlanggaran berlaku, robot itu harus bermula dari kedudukan awalnya, dan Rangkaian Neural Convolutional memerlukan latihan semula untuk mengelakkan keputusan yang sama berulang. Menawan kedudukan semasa adalah sangat penting dalam menentukan

langkah terbaik yang akan diambil. Grid Tetingkap dicadangkan untuk menyelesaikan perkara ini. Beberapa algoritma digunakan untuk membuat Grid Tetingkap untuk mengetahui kedudukan semasa robot yang bermaklumat dan relevan. Maklumat ini akan digunakan oleh Rangkaian Neural Convolutional untuk meramalkan langkah terbaik, proses itu berulang sehingga robot mencapai kedudukan sasaran.

ACKNOWLEDGEMENT

Furthering my study in the Masters level is not in the list of my life. Furthermore, I am doing this in the number one university in Malaysia and the top 100 in the world. This is something that unbelievable. This is not just about the ranking, but the professionalism and knowledge of the lecturers that are outstanding. Thank you very much University of Malaya for this golden opportunity. I was an engineer in a company and at the same time I also a student, and the challenge is when it comes to the time management. I am very thankful to my Head of Engineer, Mr. Lee Chun Siang on the support and understanding without feeling tired and bored. Sometimes, he helps in giving a thought and idea about my assignment and mini project. I will never forget the co-operation and tolerance shown. Thank you for all the supports. During the four semesters of my study, I seldom went back to my hometown, visiting my parents and family since all the classes is during weekends. Thank you, God, for giving me very understanding parents and siblings, having them as family such huge a blessing from Him. They sacrificed time and money without any hesitation and complaint. The success of mine today is actually the success of them as well. Not forgetting my supervisor for the Research Project, Ir. Dr. Chuah Joon Huang on various supports and sharing of the knowledge of Artificial Intelligence, which is now being discussed by many people. Some other times, he shared his own experience and advised me to be bolder and more creative in my work. Thank you very much. Last but not least, the Most Merciful and the Most Gracious, He eases all my journey from the starting until now, and end of my life. The blessing from Him is never-ending, and will never end. He removed all the anxiety, uncertainties and difficulties. He took away all the

obstacles that block my life. He is the one that we worship and He is the one we ask for help. Thank you for giving me a wonderful life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF DIAGRAMS

ACRONYMS

4D	four dimensional
A*	A star
A.I.	artificial intelligent
Adam	adaptive moment
CL	cubic lattice
CNN	Convolutional Neural Network
FC	Fully-connected
FCN	Fully Convolutional Neural
GPLVM	Gaussian Process Latent Variable Models
GPU	graphic processing unit
MATLAB	Matrix Laboratory
ReLU	rectified layer unit
RGB	red, green, blue
RRT*	Rapidly-exploring Random Tree Planner
SSGPLVM	Slab Gaussian Process Latent Variable Models
VLSI	Very Large Scale Integrated

CHAPTER 1

INTRODUCTION

1.1 Overview

Artificial Intelligence (A.I.) has become extremely popular among individuals in both the education or industry sectors. Their opinion on this technology is that it is the most advanced phenomenon and everybody has a strong interest to apply this technology to their own businesses especially in the robotic sector. However, A.I. itself is not only focusing in robotic engineering, but also in accounting or banking, transportation, hospitals and medicine, security and many more [15,19].

In March 2016, the world has been introduced to humanoid robot named Sophia where she can express the facial expression in 50 different ways. Moreover, she also able to answer all the interviewers' questions from the media. Sophia is the idea of Hanson Robotics, the Hong Kong based company with the collaboration of A.I. developers [9].

The gigantic social media, Facebook also has implemented A.I. in their businesses in order to solve their unstructured data management due to the uploading data from the users [23]. By implementing A.I. or deep learning, the machine allows the data to classify themselves since it becomes more sophisticated when there are a lot of data can be used for learning purposes for the machine.

The existence of robot in our daily life becomes more convenience, especially to help our home task. For example, intelligent vacuum cleaner, automatic dishwasher, these robots or machines can attach A.I. in order to improve their performance and effectiveness on

the assigned work. For this research, Convolutional Neural Network (CNN) is selected to be used among others learning program.

Simon Haykin [10] refers neural networks by the recognition from human brain which is it able to compute in entirely different ways where the human brain is highly complex, nonlinear, and works like a parallel computer in term of information processing system. Simon Haykin also provides several benefits of neural networks. The following are the benefits of neural network:

- Nonlinearity

  The brains have the capability to organize the structural constituents knows as neuron. The neuron itself can be linear or nonlinear. The neural network is made up from the interconnection of nonlinear neurons. This nonlinearity is very important character, particularly if underlying physical mechanism responsible for the generation of the input signal.

- Input-output Mapping

  Neural network is classified as supervised learning where it involves with the modification of the synaptic weights of a neural network by applying a set of labelled training samples. Each input requires the desire response or output and the network is presented with a random set, and synaptic weights of the network are modified to minimize the difference between desired response and the actual response from the given input. In another way, adjusting the synaptic weight is to reduce the error of the output. The training continues until there is not significant changes in the synaptic weight. Thus, the network is learning from the examples by constructing an input-output mapping for the given problem.

- Adaptivity

Neural networks have built-in capability to adapt their synaptic weights to changes in the surrounding environment. It can be trained to receive and react to the small changes and easily retrained in the operating environmental condition. When it operates in a dynamic environment, the synaptic weight can be changed in real time. This natural ability allows it to do pattern classification, adaptive signal processing and adaptive control.

- Evidential Response

  Neural network can provide not only information in pattern classification, but also have confidence in the decision that has made. It has capability to reject the ambiguous pattern and at the same time improve the classification performance of the network.

- Contextual Information

  Each neuron in the network is affected by the global activity of all other neurons in the network. Consequently, contextual information is dealt with naturally by the neural network.

- Fault Tolerance

  Fault tolerance potentially inherent when the neural network applied to the hardware form, capable of robust computation. For example, if a neuron or the connection link is damage, recall of a stored pattern is impaired in quality. However, due to the distributed nature of information stored in the network, the damage gas to be extensive before the overall response of the network is degraded seriously. In order to ensure the neural network is in fact fault tolerant, it may necessary to take corrective measures in designing the algorithm used to train the network.

- VLSI Implementability

  Very Large Scale Integrated (VLSI) technology is suited to implement by the neural network since it is massively parallel nature which has potential to compute in fast for the given task.

- Uniformity of Analysis and Design

  Neuron is representing an ingredient common to all neural network either is in one form or another. This commonality makes it possible to share the theories and learning algorithm in different applications of neural networks. Thus, modular networks can be built through a seamless integration of modules.

- Neurobiological Analogy

  The network of neural is motivated by the analogy of the brain where the living proof that fault tolerant parallel processing is not only fast but also powerful. Engineer interpreted neurobiology as a new idea to solve the problem more complex than those based conventional hardwired design technique.

Artificial neural network imitates human brain nerve, the way biological nervous system (like brain) process the given information. The nervous system is a combination of more than 100 billion cells called neurons, and it is composed of three major parts; cell body, soma and dendrite. Dendrite is the key element in the cell for collecting the information.

Diagram 1 represents how the neural or nerve net receive the information, perceives and define the correct decision. Receptor receives the feedback from the stimulus and convert it into electrical impulses than convey the information to the

neural net (brain). Effector then takes place converted the electrical impulses from the brain into discernible responses as system outputs.

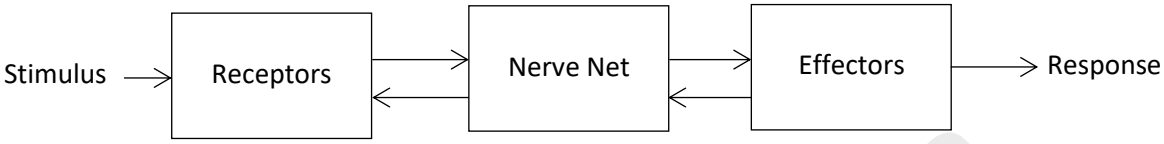

Diagram 1: Block Diagram of nervous system

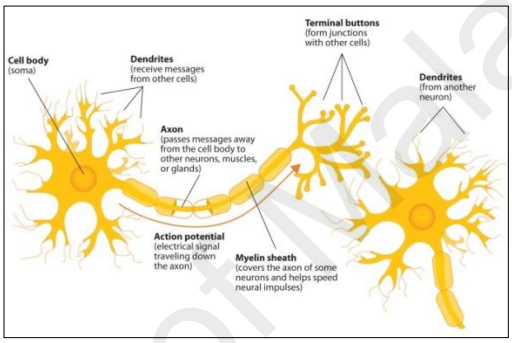

Figure 1: Neuron Component in the nerve system [7]

Dendrite will collect the information from the surrounding of neurons then pass them from soma to the terminal buttons through the axon. In simplicity, axon is a messenger which passes the information from the spinal cord to the muscles in the hands or feet or any other abdomen in the body. The passed information is in term of electrical signal.

In Figure 2, terminal buttons release some chemical into the synapses gap called neurotransmitter which relays the signals. The different terminal button is release different neurotransmitters and different dendrites are unique because they are sensitive to different neurotransmitters.

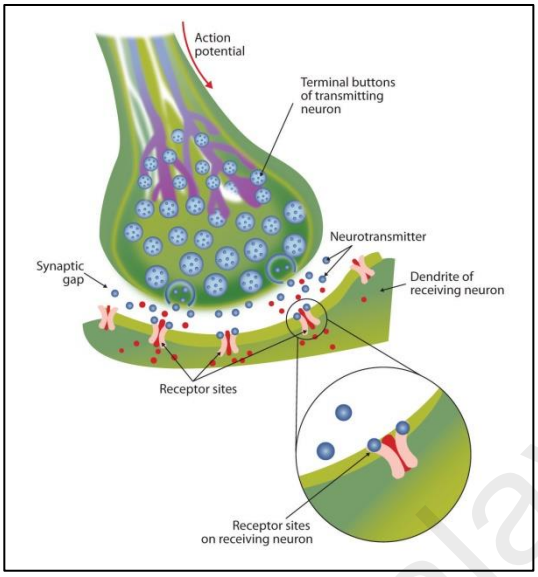

Figure 2: Terminal Buttons emit the chemical to other neuron's dendrites [7]

In engineering learning, neural network is a kind of supervised learning. It's requires example to learn in order to perform the given task. There is a multiple type of module available in neural network called Convolutional Neural Network (CNN). Convolutional Neural Network (CNN) is one of the modules in deep learning which is commonly used to solve artificial intelligence problems [17] especially in pattern recognition and machine learning [17] as the benefits of it is it able to extract the multilevel features from the sensors [17].

The architecture of CNN builds from multiple layer of perceptron, one layer of input, one layer of output and has multiple layers of hidden layers.

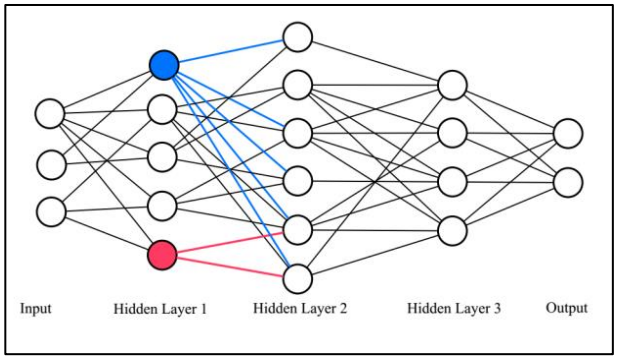

Figure 3: Simple Architecture of Multiple Layer of Neural Network

For CNN the input layer is composed of the numbers of the array of pixel from the given input picture. But the size of the pixel array depends on the class of the input. If the input is RGB, then the pixel array is 32x32x3 where three is representing RGB. If the input image is a colour image, the size of the array is 480x480x3 [18,23].

Second layer which is called hidden layer is always Convolutional layer. The input of this layer is the pixel array from the input layer [20]. The filter (biologically so call neuron/ kernel) is the element use to extract the data from the input which been called weight or parameters. The filter will slide across all over the input area and this activity called convolving. The size of the filter is called receptive field. To allow math working fine, the depth of the image should be the same with the depth of input. During convolution of the input, every location in the input will be located in a new number and those complete combinations of the input will produce an activation map or feature map [22].

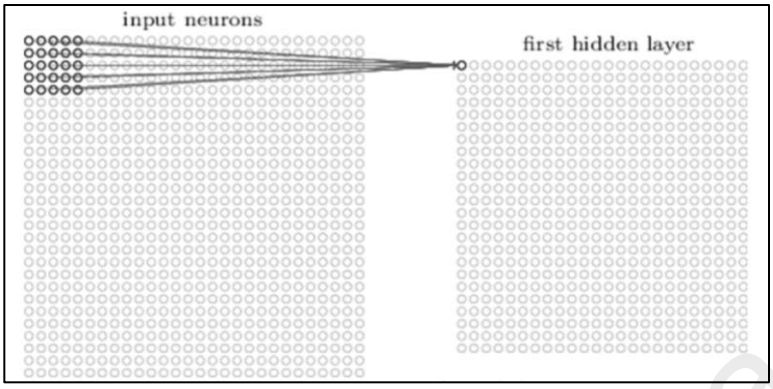

Figure 4: Left is feature map

Then, this activation map of the first Convolutional layer will be the input to the second Convolutional layer and the same process is repeated on the second Convolutional layer. The activation map of second Convolutional layer represents higher level features. When the network goes through more Convolutional layer, it will create more activation maps with more complex features.

The layer that receives the high-level features as an input is called Fully-connected layer. Basically, this layer will produce the output at the N-dimensional vector where N is the number of classes that the network has to choose from. Classes are referring to the output feature type. For example, if the network works for numbers classification from zero to nine, then the classes should have 10 where it is from 0 to 9. But each number in this N- dimensional vector represents the probability of a certain class. For example, the resulting vector is [0 .1 .1 .75 0 0 0 0 0 .05] represent 10% image of 0, 10% might represent image of 1, 75% probability that image of 2 and so on. In simple way, Fully-connected layer used to determine which features are the most correlated to which class and has certain weights so that the network might has correct probabilities for each different class.

When a baby is born, there are many things that he or she needs to learn in order to be a grown human being. For example, they need to learn how to feed themselves, how to wear cloth, how to grasp the pencil properly. All those activities require training to achieve perfect condition. Same goes to artificial neural network which require training to become as perfect as possible in analyzing the input and produce the correct and precise output. The way how the network does the training is by adjusting the filter weights through the process called backpropagation.

Backpropagation can be classified into four groups which is:

- Forward pass
- Loss function
- Backward pass
- Weight update

During forward pass, the network receives the input and straightforward to find the actual output. It is like natural forward direction which flow from the input into the system and directly gives the output.

Loss propagation is when the desired output meets the difference with the trained or actual output. The network is understood when the prediction is smaller than the desired output (giving positive value) moreover, it becomes overshoot when the prediction is over the actual value.

After determining the losses, backward pass is required to perform in determining which weight contributed most to the loss, then adjust the weight of them so the loss could decrease.

Weight update is the last section in backpropagation processes which it takes all the filter weights and update so they change the direction of the gradient. The new weight is depending on the learning rate where it is determined by the creator of the network. High learning rate produces bigger gap, and the time require to update is less, and vice versa happened when the learning rate is smaller. Once the training is finished from forward to weight update, it was called one training iteration. The program will repeat the process for fix number of iterations of each set of training inputs.

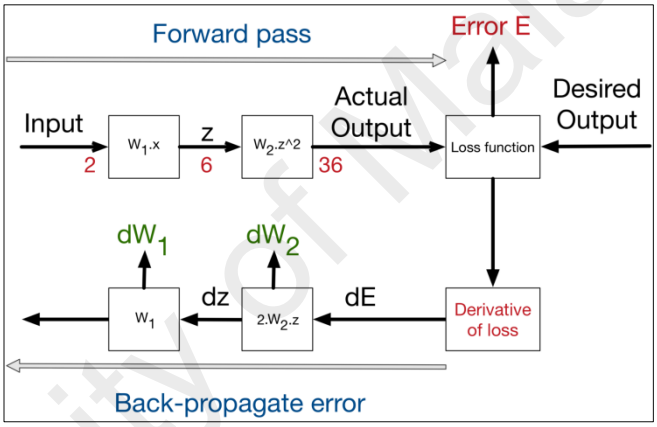

Figure 5: Backpropagation Process flow from input to backward pass the error

After the training is completed, the network has the weight of the filter that has been tuned correctly and it's able to work fine with the other new input batch.

The other main advantage of CNN is it's the best among deep-learning neural network where it's easy to train and has better generalization on the Fully-connected layer compared to another network [27]. Some other times, people used the hybrid system where the combination of CNN with another algorithm. The advantage with hybrid system is the loss accuracy is minimum and faster in calculation [25].

The idea on the training of the network is from back propagation [15] by computing the loss respected to the network parameters.

1.2 Problem Statement

The implementation of the robot becomes wider nowadays. It's not only focuses in the manufacturing industry, but it also starts in other kind of industry and with the different purposes. Like in this research, it is focusing in zero human interaction with safety as a compliment.

There are several lists that should be achieved:

- To design a control system which allows the robot available to work robustly despite the limited knowledge of the environment. In the other words, it is desired to design of obstacle avoidance behavior for robot which is effective in face of unforeseen static obstacle configurations.
- To execute several specific behaviors simultaneously such as static obstacle avoidance and goal searching behavior.

1.3 Objective of Study

The objective of the research is

- To use Convolutional Neural Network to control robot system in obstacle avoidance behavior
- To design a static obstacle avoidance system
- To design and develop a mobile robot simulation using MATLAB programming language

1.4 Purpose of Study

The purpose of this research being done is due to improve efficiency of human behavior by applying artificial intelligence to the mechanism/ robot. The robot may face several iterations of training in order to achieve optimum result. The research is done by using MATLAB in order to execute and examining the behavior of the robot.

1.5 Scope of Project

The boundary is being set to this project are:

- Develop the robot simulation replicate the actual environment faced by the robot.
- Apply Convolutional Neural Network on the robot to achieve the objectives.
- Allow the robot to avoid the static obstacle from the applied A.I. algorithm.

1.6 Outline of Project

This research project consists of five chapters. Starting from Chapter 1 consists of the introduction and the background of the research, problem statement, purpose and scope of the study. Continue with the next chapter, Chapter 2 involves on the research or project that have been done by the other researchers, examine and find the idea and improve their problem stated in their research.

Next, Chapter 3 is about the way or method that using during the experiment in order to achieve the objectives. It contains about how and why the experiment is conducted and gives the reflect to the research goal. After finish the methodology

side, continue with Chapter 4 which is the result finding and the observation analysis from the experiment that have been done. From that the discussion is included and the result is elaborated.

Last but not least, Chapter 5 consists of the conclusion throughout the research and the details of problem that should be overcome on the next project.

CHAPTER 2

LITERITURE REVIEW

2.1 Introduction

This chapter discusses about the nature of path planning by using CNN from the previous study. In other hand, the discussion also includes how CNN is work for obstacle avoidance when it is in static position. Last but not least, the final part of the chapter will conclude all the previous study has made based on the overall content of the chapter.

The purpose of path planner is to calculate an optimum possible path from designated origin position to the destination position. The primary concern of planning the path is free from any collision and defining the optimal path to the goal position.

2.2 Path Planning of Mobile Robot

Andon T. Z. and Ignat R.E. (2018) mentioning that the path planning in Convolutional Neural Network facing several layers from input layer to the output layer. In his paper, he used the combination of Convolutional layers and max-pooling layer which being called multilayers Convolutional neural network to plan the path of the robot. In addition, convolution kernel with different scales are produced from sparse auto encoder training algorithm. Parameter of hidden layer is a series of convolutional kernel will later be used to extract the first layer features. Next, second layer features input will be extracted from max-pooling operators where it can improve the invariance of features or in another word minimizing the error. The last

layer is fully connected layer before the output layer is used to define the path planning task.

Navigation is one of the problems that is under research for the decade. The success of navigation process requires four building blocks of navigation which are perception, where the robot should be interpret its sensor to correct meaningful data; localization, the robot should determine its position in the environment; cognition, robot should be able to decide how to achieve the target; and motion control, the robot must be able to modulate its motor outputs to achieve the desired trajectory.

Punarjay C., Klaas K., Tom R., Stijn W., Tinne T. and Luc V. E. (2017) were presented indifferently from the other research. They were implementing the CNN in the flying object like drone which is called quadrotor. This kind of light-weight quadrotor UAC's is prone to has crashed because of it flying a little autonomous capability. It becomes worse when it was flown in indoor environments.

The task of planning the path is important to fulfil the criterion of full backpropagation in order to make is working fine. Moreover, that allows the network to train properly an underlying cost function network from the given input data (Eike R., Jannic Q., Christoph S. (2017)). This is important for correcting the filter weight in order to reduce the error and giving the precise result. Eike R., Jannic Q., Christoph S. (2017) were using state grid for planning execution where the state grid is working as an equidistant discretization of the state space.

Lei T., Shaohua L., Ming L. (2016) stated that their research requires visual information which is as RGB information as their input and the output is the moving direction of TurtleBot. The feature map of their network is from the Gaussian process

latent variable model which purposely for proves the effectiveness of the network. The method proposed by the writers is using Deep Q-Learning where the deep neural network is used reinforcement leaning type on how the network is being trained. Usually the neural network trained by supervised learning.

The objective of the path planner defines by Valeri K. and Jianli Y. (2011) is to calculate a path from a given starting point of the vehicle to the destination position and the main attention to the planning is to avoid the collision with the obstacle. Another concern is to compute a realizable and finding the most optimal path possible to reach the goal location.

The path planning requires input and there are several types of image properties used by the researcher and different method has been used to prepare the input data. Changlong L., Bin Z., Chunyang W., Yongting Z., Shun F., Haochen L. (2017) taking their input from the raw images which were obtained from the camera, then converted them into raw pixels purposely for steering command. The input images were in RGB properties to allow the robot move straight, turn to right or turn to left.

Punarjay C., Klaas K., Tom R., Stijn W., Tinne T., and Luc V. E. (2017), stated their input of the CNN is RGB images where they trained the network by using 260000 of images. All the dataset is used supervised learning and this dataset is obtained by online and offline.

The writers mentioned the CNN is used to predict the depth maps of a given input. The Behavior Arbitration control algorithm is used to have an input of depth map meanwhile the output is angular velocity ϕ. Yaw and pitch axes are used to steer the

quadrotor fly away from the obstacle and finding the minimal path to the given destination point.

Glasius R., Komoda A., and Gielen S. (1994) were introduced the path planning system which consist of a large number of identical processing units called neurons. All these neurons are in position of d-dimensional cubic lattice (CL). A number of neurons in the network are connected only by their neighbours by excitatory and symmetric connection. The writers make an assumption of lattice which it was represented to a topologically ordered map where it can be obtained by learning type of Kohenen.

The advantage of this map is as a representation of the robot configuration space with nodes homogeneously distributed call neuronal space. Each neuron in the working space was corresponding to a certain subset by receptive field. The state variable of each neuron can be changed because they were affected by the other neurons and due to the external sensory input. The synaptic connection between each node will experience excitatory and symmetric phenomenon.

2.3 Convolutional Neural Network (CNN) in Path Planning

To do path planning in a state grid commonly modelled as a graph where every edge of the graph is represented by neighbouring cells in the grid (Eike R., Jannic Q., Christoph S. (2017)) by using the Dirac Delta Function. The Dirac Delta Function works as shifting by a, $\delta\ (t\text{-}a)$ convolved by the function of $g\ (t)$ and the result is the function of g shifted by a, $g\ (t\text{-}a)$. The same ways apply to the discrete convolution. One shift of one state in a grid can be modelled as convolution with a two-

dimensional transition filter mask. In simplicity, any transition in a grid can be modelled as Convolutional layer with known filter mask.

Value Iteration Module is used to calculate the minimum cost per state within a grid map from a given starting point. But it does not compute the optimal path from stating state to the goal state. There are several steps provided by Eike R., Jannic Q., Christoph S. (2017) which involved in this process. The following steps are:

- Initialization

  Every cost grid needs to initialize with arbitrary in very large amount in every state. All the cell at stating state will be set to zero.

- Non-zero padding

  Pad the entire state grid with arbitrary in very large amount. Zero padding introduces faulty results in the border area.

- Cost propagation

  Convolve the cost grid with transition filter to execute all possible transitions.

- Cost accumulation

  Add cost per state and transition as additive layer.

- Assign minimum cost

  Min-pooling used to compute minimum cost per state over transition direction of the cost grid.

- Recurse

  Repeat above process until the convergence is reached.

Eike R., Jannic Q., Christoph S. (2017) stated path evaluation module was used to compute the optimal path starting from the starting state to the goal state. The several steps are required like following:

- Initialization

  Set all zeros as initialization for all state grid.

  Set cell destination into one. The transition filter is flipped to mirror around the centre point and input, outputs direction is exchanged.

- Transition selection

  Do multiplication of the current state grid with transition selection mask. Then the output will become a grid with single lube in the cheapest possible transition in the current state.

- Zero padding

  Pad the state grid zeros and the state now is non-occupied states.

- Propagate state

  Convolve with the flipped transition filters.

- Recurse

  Use the result of propagation state for the new iteration until starting state is reached.

For forward pass of CNN, the mapping can be done by dimension of W×H×1 to W×H×K where K is possible transition.

Lei T., Shaohua L., Ming L. (2016) describe three operations of the forward process in CNN is:

- Convolutional

Work as image filtering by taking summation of weight of pixel values in the receptive field. The smaller receptive field the smaller classification error produced.

$$y_{ijk} = (W_i * x)_{jk} + b_i$$

where,

y_{ijk} = pixel value of coordinate j,k of the i^{th} output of feature map.

W_i = i^{th} convolutional kernal

x = input

b_i = i^{th} element of bias vector which correspond to the i^{th} convolutional kernel

- Nonlinear activation

Non-linear function is applied to the output of the feature maps by using sigmoid function $s(x)=1/(1+ e^{-x})$ and the hyperbolic tangent function $\tanh(x) = (e^{x}-e^{-x})/(e^{x}+e^{-x})$. Later rectifier is applied $f(x) = \max(0,x)$.

- Pooling

Pooling process taking the maximum or average value of image patch to improve the robustness of the network and reduce the noise observation effect.

- Stride

It exists in Convolutional layer and pooling layer. When the value of slide greater than 1, the output of feature maps is down sampled by factor of s. Stride is useful to reduce the size of the whole network.

The writers provide several stages represent the hidden layer of the network. Stage 1 includes Convolutional layer where there are 32 convolutional kernels with size 5x5, and another layer is ReLU layer where 2x2 pooling layer with stride 2. Meanwhile in Stage 2 contains of Convolutional + Activation + Pooling Layer and the size is same like in Stage 1. For Stage 3, the writers present 64 convolution kernels with size 5x5, 64 feature maps with 20x15 size, and the last one fully-connected layer contains of 5 nodes where they are represented scores of each output state. Lastly, the final decision is being calculated by applying soft-max function to score of 5 possible states.

The last layer is visualized of feature maps are adopted from GPLVM, Spike and Slab Gaussian Process Latent Variable Models (SSGPLVM) into lower dimensional space.

Valeri K. and Jianli Y. (2011) have discussed about two-dimensional known environment by considering the stationary of obstacle which is in polygons or ovals.

Both of the proposed algorithm is based on the potential field methods where the algorithm solves using local minimum problems and generates the optimal path in relatively small numbers of calculations. There are several approaches described by the writers in order to choose suitable algorithm and contributing several advantages and avoiding the disadvantage.

Valeri K. and Jianli Y. (2011) using potential field method algorithm where the basic concept is to fill the workspace with an artificial potential field in which the robot is attracted to the goal position being at the same time repulsed away from the obstacles.

To avoid fail in finding the free path and trapped in local minima, the writer comes with some properties.

i. The magnitude of the potential field should be unbounded near obstacle boundaries and must lower within range.
ii. The potential should have a spherical far away from the obstacles.
iii. The equipotential surface near an obstacle should have a shape similar to that of obstacle surface.
iv. The gradient of potential and their effect should be on the path must be spatially continuous.

The proposed algorithm contains several advantages where: -

i. The descriptions of the obstacles are implemented directly in the simulated annealing neural network.
ii. No need to perform learning of the workspace off-line using backpropagation.
iii. There are no other additional calculations and processing when obstacles are added or removed.
iv. Develop the simplest solution of the local minimum problem and generated path are conditionally optimized in the sense that they are piecewise linear with directions changing at the corners of the obstacles.
v. Provides parallel calculation where improves the calculation time and less memory consume.

The writers described the state of path on energy function where

$$E = wlEl + wcEc \tag{2.1}$$

where w_l and w_c is the weight.

The particles or robot would face this repetitive process of planning the path with collision free like in the Diagram 2.

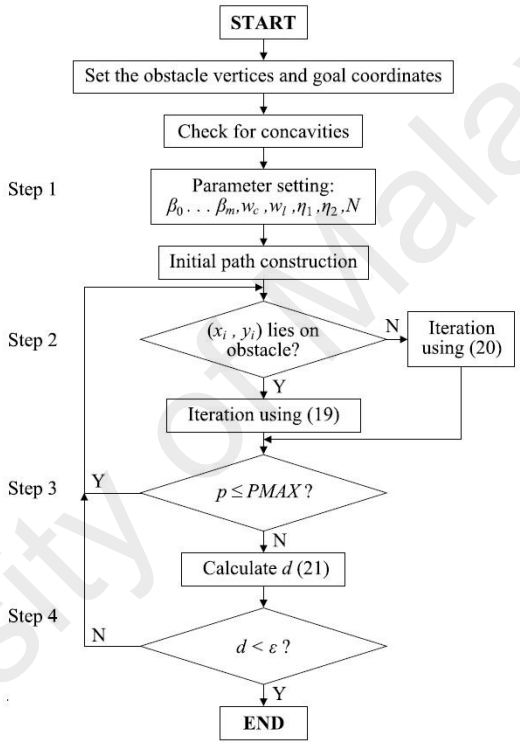

Diagram 2: Block Diagram of free collision path planning

Step 1 is initialized for parameter setting of weight, numbers of vertices, adaptation gain. Step 2 determining if the path is created inside the obstacle will perform iteration 19 while free collision path will continue the calculation with goal with focusing on minimizing the path. Step 3 is to perform a number of calculations on Step 2 where p is defined by the programmer. The last step is Step 4 which used to test the convergence d of the path lengths at time t and time (t+p). The process is

kept repeating to Step 2 until the difference of path lengths is less than a constant, in the other hand, the robot reaches the goal position.

The properties of input that given to the network might take some amount of space that will slow down the process. Yang L., Shujuan Y., Yurong L., Yuling J. (2016) has turn the input image into grey and change it to 112 × 112, also divide it by 255 in order to normalize it as input data. The writers set the first feature image in Convolutional layer are 104 × 104, number of second feature image in Convolutional layer is 48 × 48 and third feature image in Convolutional layer is 20 × 20 to optimize the number if Convolutional layer feature image.

Convolutional layers are the layer that used to extract the image properties, reminiscent of simple and complex cells in the primary visual cortex, while max-pooling layer is used to substitute the subsampling layer and back propagation to train the CNN. The writers also include two stages of feed forward pass and back propagation into the network.

The writers described the purpose of every layer that they used to build the success network and there are include Convolutional layer, max-pooling layer and classification layer.

In CNN, the parameters of each convolutional kernel are trained by back propagation algorithm where they were used to extract the difference of features input. The initial Convolutional layer will obtain low-level features. For example, line, corners, edges, and etc. Meanwhile the other convolutional kernel will use to extract the higher level of features. Every output map feature was combined multiple input maps with

convolutions. Sigmoid function or Tanh function with the additive of bias is used in details implementation of the network. Both bias and kernel weight were trained by using supervised learning approaches.

The purpose of Max-pooling layer recognize by the writer is used to substitute a sub-sampling layer in the network and the function of this layer is used to minimize the variance and finding the maximum value of particular feature over a region of image. For example, it will ensure that the result obtains is the same for the image features that have small translation. This is a very important layer for object classification and detection. The benefit of this layer is not only faster in convergence, but also improves the generalization ability when processing the superior invariant features.

Classification layer proposed by the writers used supervised learning in order to train the network. It's composed of multiple classifications for path planning detection at different locations. The maximized activation neuron is representing the best path planning. Another function of this layer is visualizing the image feature. It takes the average image patches that are related to neurons with maximum response in the upper layer.

Basic understanding of Local Dynamic Path Planning is when the robot needs to start from one position to the goal, the robot is able to reach the destination and avoiding the obstacle by feasible shortest reference path based on machine vision information. Reference path is the sequence of points where there are exists certain distance between them and at the same time this reference path must satisfy the dynamic avoidance condition. When the robot was in motion along the reference path with a certain speed, a new round of path planning is performed. The reference path is

replaced by new one and this iteration will make the robot move near to the goal. In this case, the local path planning should be long enough to allow the robot plan the path in the forward un-planning area and create two planning path that have long enough overlap length to generate reliable and optimal path.

The writers put assumption on their project:

1. The robot moves in finite dimensional space
2. The boundary of the obstacles in the X and Y direction
3. Obstacles in the working space are described by convex polygon.

The 4 × 4 transformation matrix is used to calculate the movement of the robot.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} A & B & C & L \\ D & E & F & M \\ G & H & I & N \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.2}$$

Where A, B, C, D, E, F, G, H, I is represented the coordinate of rotation, L, M, N represent the moving part. *x, y, z* is represented the robot position itself. The changes of robot position can be calculated by using this following matrix.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & L \\ 0 & 1 & 0 & M \\ 0 & 0 & 1 & N \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x+L \\ y+M \\ z+N \\ 1 \end{bmatrix} \tag{2.3}$$

Hachour O. (2011) provides general structure of the neural network navigation like follow:

Knowledge mapping: how model of external environment plays important role in the intelligent robot behavior. How the neural network work efficiently imitates like human brain with parallel computerization.

Action: different map sensory information is classified in several vectors where each of them responds to particular action.

Reinforcement learning: this kind of learning allows to associate between detection sensory situation and appropriate action taken. These associations are formed in unsupervised manner; with no supervisor or teacher require.

The writer proposed the motion action taken by the robot it four ways, either moving to the front, turn to left, turn to right and turn back. When it represents inside matrix form $A = [A_F\ A_L\ A_R\ A_B]$. In the other hand the static target localization is defined by $T = [T_F\ T_L\ T_R\ T_B]$ while static obstacle avoidance situation is defined by $O = [O_F\ O_L\ O_R\ O_B]$. The layer composed in the network consists of three layers. Layer 1 is representing input layer with four nodes receiving the component vector of T. Layer 2 is receiving the output from Layer 1 as its input. Layer 2 is where hidden layer in computing the feed-forward propagation with Sigmoid function. The learning rate set by the writer is in between 0 and 1. The last layer is Layer 3 which is represented an output layer where determine the position of the robot. The back propagation is used in order to train the robot adapting the environment with free from collision without losing its main target.

Changlong L., Bin Z., Chunyang W., Yongting Z., Shun F., Haochen L. (2017) provide five Convolutional layers and three fully-connected layers. The working

framework using by them is Caffe which use to extract and learn feature from the input dataset. Caffe is the deep learning open source frame which is commonly used in data extracting for the machine vision. The system of the robot is performing the instructions which were predicted by the CNN based Robot Operating System (ROS) where ROS is an open source framework for robot where it can provide a similar operating system for heterogeneous computer clusters.

In Figure 6, the ROS_Caffe works as a bridge to provide communication between Caffe and ROS. ROS requires raw image will be later trained by CNN model in making the decision for robot to plan its path.

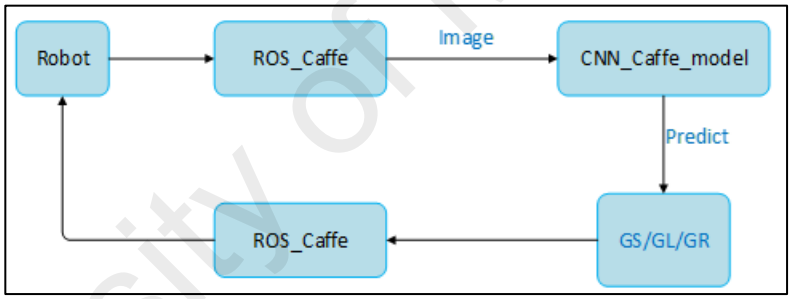

Figure 6: The flow chart of running

Keith S., and Wallace L. (2016) also taking RGB images as the input to the CNN to control the robot with alternating between Convolutional layers and max-pooling layers followed by fully-connected layer. The geometric of environment is being extracted by using Convolutional and Max-pooling layers while Fully-connected layers works as general Classifier. The network is presented like in Diagram 3.

Diagram 3: Convolutional Neural Network

Each layer provides different filter size and kernel size. Conv1 is the first convolution layer comes with 75 filters with size of 10×10, and stride of 10. Pool1 is the first max-pooling layer with kernel size of 3×3 and stride of 2. The layer then continues with Conv2, the second convolution layer comes with 75 filters with the size of 5×5 and stride of 1. The output of this layer becomes the input of Pool2 where it is the max-pooling layer by 2×2 kernel size and stride of 2. Convolution process is next taking place by Conv3 where the size of its filter is 75 with 5×5 stride of 1. The network then has alternating Fully-connected Layer (FC1, FC2) and Rectified Linear Unit (ReLU1, ReLU2) before the last layer, Softmax Layer comes with three outputs.

The input maps of the Convolutional layers are using 2D convolution with a rectangular filter. The result of convolutions is summed and transformed by a hyperbolic tangent activation function when the previous layer contains more than one map. Higher activation occurs where the filter better matches the content of the map. The max-pooling layer has the output where it is the maximum activation function of non-overlapping square region of the layer's input. This layer will be select the winning neuron. The function of the last layer, Softmax layer is to calculate the activation of each three classes where it can be translated as robot control.

Keith S., and Wallace L. (2016) define their network with having no pre-training, the weights were randomly initialized, adaptive gradient descent is used to minimize the loss over the training set.

The robot is moved mannerly will enter training mode to update the CNN when it senses the existence of an obstacle. After updating CNN, the procedure begins anew

with better performance. By this way, the robot will be able to bootstrap obstacle avoidance in any environment.

The CNN modelled by Punarjay C., Klaas K., Tom R., Stijn W., Tinne T., and Luc V. E. (2017) is based on Global Coarse Scale Network where it builds in sequence of Convolutional layers alternated by max-pooling layers and ReLU layers. The network then followed by two full-connected layers.

Each of the Convolutional layer will attach with a batch normalization layer to help against overfitting by normalizing data batches processed in particular layer to a Gaussian distribution.

The scale invariant error L (y, y^*) in log space is used to locate the network loss function which is happened between estimated depth image y and ground truth depth image y^*. Some algorithm also attached to the system to define the relative depth of different obstacles.

Glasius R., Komoda A., and Gielen S. (1994) define each of the neuron should be initialize their position to zero. Due to the influence from external factors, which clamp the neuron at the target nodes, thus the value of "one" and all occupied nodes clamped to "zero".

In Figure 7, the writers were mentioned all neurons in the network were connected bi-directionally to the nearest neuron within a distance. The activity of the neurons starts with the closest one to the target position clamped to value "one" and all activity of neuron that correspond to the obstacle are clamped to value "zero". The neuron will face update of each of their activities in parallel at every time-step.

Each position of the robot is coordinated by x and y in 2-dimensional workspace. Each of them is corresponding to a small unit area in the discretization of the workspace and neighbouring area also corresponds by neighbouring neurons.

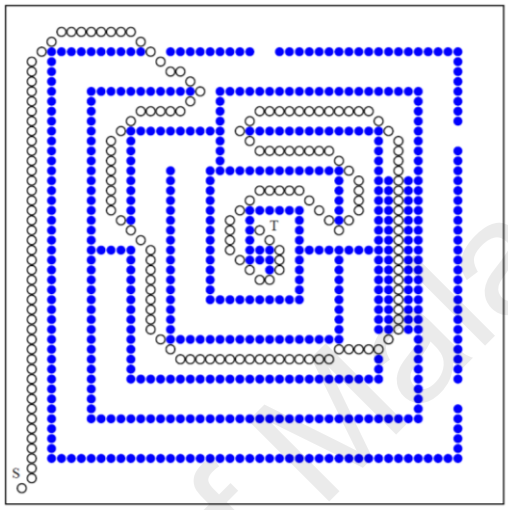

Figure 7: Blue dots are representing the wall or obstacle and the white dots is the path of the robot

S which stated in the Figure 8 is the start position while T is the target position. If a continuous path exists between the target and the initial position, the shortest path length has been chosen.

The output of this Figure 8 architectural neural will attached to linear activation function. Aleksandr I.P., Komstantin S. Y. and Roman S. (2017) provide 8 types of output from by the system. The network initially has Convolutional layers with sigmoid or ReLU activation function. The objective is to construct the invariant representation of the pattern occurred. This convolution layer has 3×3 filter size with stride of 1. Next followed by fully connected layer where it accepts flatten output after the convolution layers. Dense layer in Figure 3 is used tanh or ReLU activation function and 50 neurons in hidden parts.

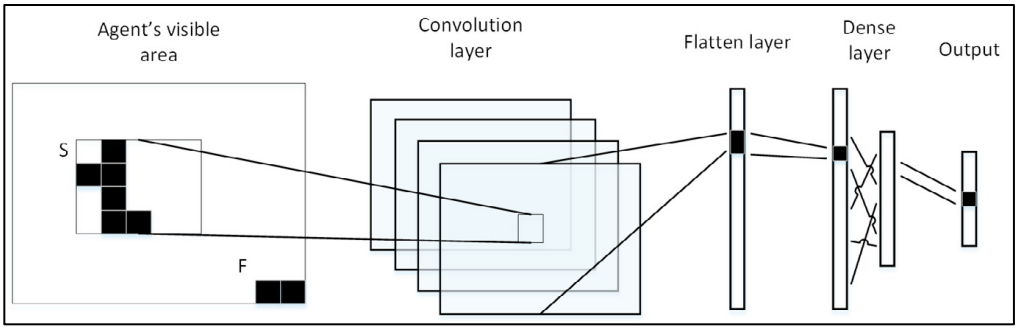

Figure 8: The network used by the writer

Lei T., Shaohua L. and Ming L (2016) implement path planning optimization by the CNN in indoor environment. The CNN used to determine the set of high-dimensional features map which later becomes the input to the fully-connected layers. The complete network diagram is in Figure 9.

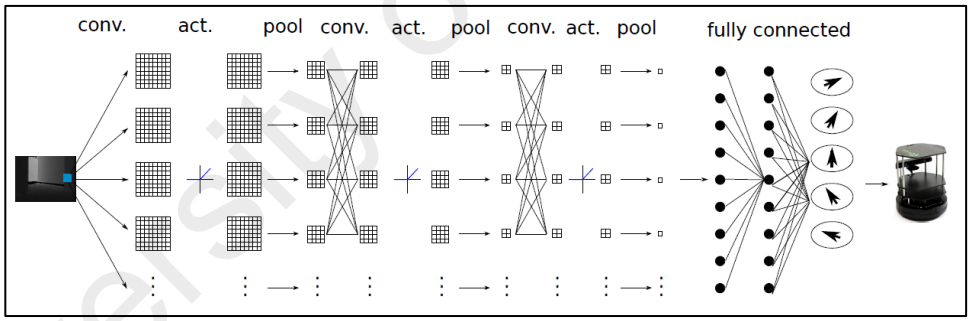

Figure 9: Complete network architecture presented by Lei T., Shaohua L. and Ming L

The writers also include back-propagation in the research to allow learning process happened in the network. Three stages compromised by the writer and they are convolution layer, non-linear activation layer and pooling layer for feature hierarchy stage.

Convolution layer works as image filtering. It works by taking the sum of weight of pixel values in a receptive field. It proves that large receptive field may contribute to

the classification error. The step over pixels is controlled by stride number. It is able to reduce the whole network parameter size.

The non-linear activation function layer is reacting as an element-wise non-linear function applied to the output feature maps where this imitate by the biological nervous system, to process of stimuli transmitted by neurons. Commonly Sigmoid function, Hyperbolic tangent function (tanh) function is firstly used for the activation. Later a piecewise linear function called rectifier is being applied, ReLU. The property of ReLU is it can execute faster than the previous two non-linear functions for activation that mentioned before.

The network then followed by pooling layer which takes the maximum or average value in an image patch or input. The objective of this layer is to improve the robustness of a network and reduce the effect of noise observation.

The input of the network is the depth maps because it provides straightforward information of traversability. The output of the network has to experience two approaches to create the decision that has to be made; control command discretization and confidence-based regression.

Control commands discretization can be generated by linear classifier and it's been done by soft-max classifier. The output control commands are samples and discretized to allow the robot to reach the destination. The output of soft-max classifier is taken as the probability of each discretized direction in the Confidence-based regression. It solves the shortcomings of a winner-take-all strategy. When the

system is confused in making decision, it can be solved by linear combination of the decision candidate.

Besides, rather than using the graph data, mainly CNN is the best in image processing. Lingyan, R., Yanning, Z., Qilin, Z., & Tao, Y. (2017) extracted the training data for their network by using spherical camera which used to capture the surrounding image. The camera is mounted on the robot 1.9m from the ground.

From Figure 10, the uncalibrated spherical images had fed to the first layer of Convolutional layer, and continue with Pooling layer. The output of the network is the navigation signal which contributed to the steering direction and angles of the wheel. In order to have effective training, the network should consist of a balance training data set.

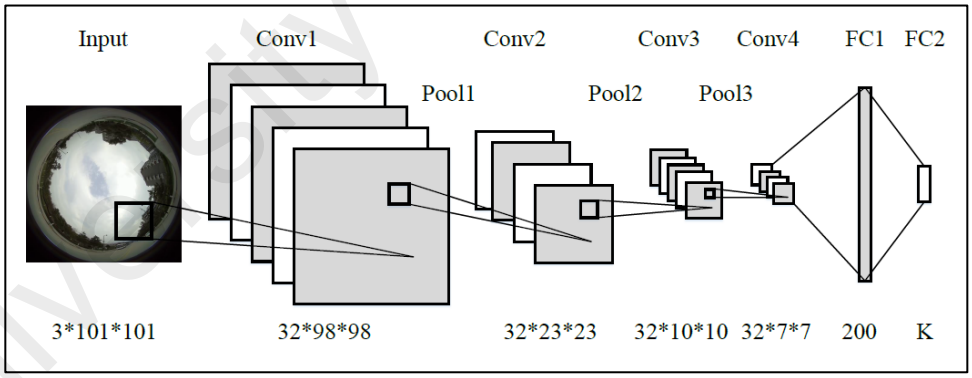

Figure 10: Network Architecture used by L. Ran, Y. Zhang, Q. Zhang, T. Yang

Another application from autonomous vehicle, CNN beneficial in rescue application and home cleaning [19]. Lei T., Shaohua L., Ming L. (2016) explain in detail each of the layers consist in the Convolutional Neural Network. The writers stated that in the Convolution Layer, the images are experiencing image filtering, which takes the

sum of the weight of the pixel values in the receptive field. The larger the receptive field the larger is the classification error. The output of this layer is defined by pixel value of the specific coordinate is equal to the result of the feature map convolve with the input, and later will be added together with the bias vector.

After the inputs are convolve, they should apply to the element-wise nonlinear function. This is where the process of stimuli transmitted by neurons. Either the output of feature maps applied by sigmoid function, or hyperbolic tangent function, or tanh function. After that it continues with the rectification of the input. This process is done in the ReLU layer. For Pooling Layer, the process is taking the maximum or average in the image patch. It aims to improve the robustness of the network and reduce the noise observation. Some of the layers contain stride where it is the step over pixel during the image convolution.

In Figure 11, the writers input the depth maps since it provides the straightforward information to the network, continued by soft-max classifier for classification process. The writers come by 2 approaches in decision making of the output.

The first approach is generated by linear classifier. Soft-max classifier is being used and the output command is sampled and discretized. Each label that's being used is representing the specific categories. For exploration task, the robot should generate the control command where the writer should adapt some trade-off. In the output state space, the speed and turning of the wheel are continuous, therefore this situation should be sampled and discretized. The state should not be too few and not to be too many to avoid the inaccurate result. The discretized step for the control command is

0 for turning full right, 1 is turning half right, 2 is moving straight forward, 3 is turning half left, 4 is turning full left.

Second approach stated by the writers is adopting the confidence-based decision-making strategy by using same soft-max classifier. Different with the first approach, second approach work by the winner-take-all strategy to make the decision, called confidence-based approach. The agent's confidence making the decision referred to the output of the soft-max classifier.

The training data is collected from the indoor depth map and sampled into 5 categories of the desired direction. The data size is 160×120 is one over quarter from the original size in order to reduce the computational cost.

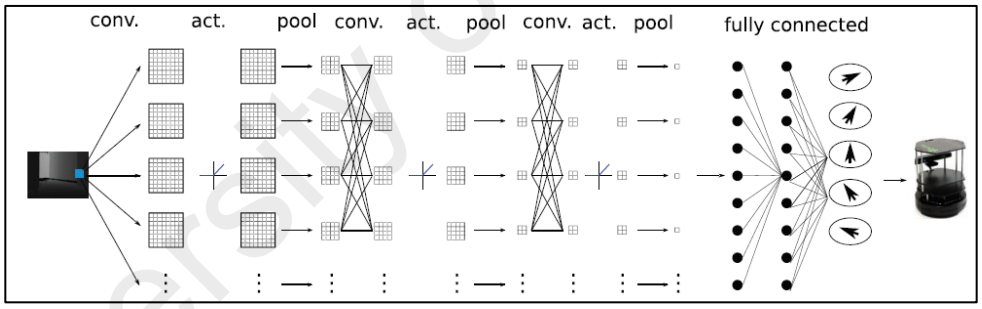

Figure 11: The network architecture proposed by L. Tai, S. Li, M. Liu

Preparing the data which use to train the network keep emphasizing to get the accurate output result. Training data can be in RGB or gray image. Yang, L., Shujuan, Y., Yurong, L., & Yuling, J. (2016) reduce the amount of data without losing the details of the image by changing the properties of image into grey and resize the image into 120×120. Later normalize the image by dividing the size by 255.

The writer did some experiment to define the optimum number of feature image in the Convolutional layer. From that the writers conclude that in the first Convolutional layer, the feature map is 104×104, feature map in second Convolutional layer is 48×48 and third convolutional feature's map is 20×20.

Back Propagation is use in the network which include feed-forward pass and back propagation pass.

There are three assumptions made by the writers about the robot working space. First is the robot should be able to move in finite directional space. Second, the extension in X and Y direction can be considered the robot as particles, and the last one is the obstacle located in the working space can be described by convex polygon.

No'e. P.-H., Fernando C., and Luis M. (2018) proposed a method of learning from demonstration strategy formulated as classification task where the Fully Convolutional Network (FCN) used to learn in order to plan a path to the goal by supervised way. This strategy combines with an optimal path planner to guarantee free collision and work in effective mission.

Combination of Fully Convolutional Network and the Rapidly-exploring Random Tree Planner (RRT*) effectively planning the navigation task and to avoid collision or wrong predictions. The input data is collected from the laser scan sensor together with detection of human location and direction. The point-clouds of data sensor are projected in 2D grid of 200×200 pixels. The orientation of people is marked as a circle and triangle in the point clouds, and the goal location is marked as small circle.

The working grid is used as grey-scale image with black background and each of the elements listed before takes different grey intensity.

A few decades the autonomous vehicle is only implemented in manufacturing industry. This application is helpful to pick and place the product or move the product from one location to another location. To widen the technology, people nowadays come out by self-driving car with artificial intelligence. Many car manufacturers start to implement this kind of business.

Mariusz B., Davide D.T., Daniel D., Bernhard F., Beat F., Prasoon G., Lawrence D. J., Mathew M., Urs M., Jiakai Z., Xin Z., Jake Z. (2016) were testing the self-driving cars using CNN with images captured from front camera to command the steering direction. The car able to drive in condition where there is with road lane marking or without lane marking and can be operated under unclear visual guidance like in the parking lot and unpaved roads.

The writers fed the image into CNN and steering command is the output of the network. Then the command needs to compare with desired command and the weights of CNN adjusted so the command will bring closer to the desired output. The weight adjusted done by back propagation process.

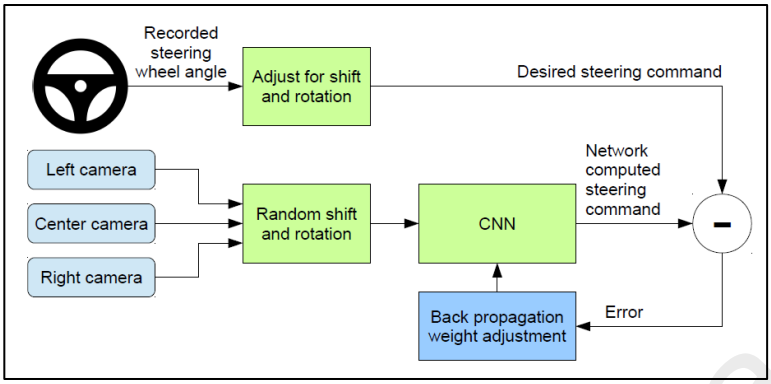

Figure 12: Block Diagram on how the self-driving car trained by the network

Shichao Y., Sandeep K., Chen M., Stephanie R., Manuela V. and Sebastian S. (2017) were predicting the path trajectory from one single image which contains ground-truth label. The CNN network has 2 stages where the first stage is to predict the depth and normal surface of the input image and the second stage is to predict the possible path for steering command.

2.4 Convolutional Neural Network in Obstacle Avoidance

Typically, before the implementation of neural network for obstacle avoidance, the robots accomplish the objective by using ranging sensor and many algorithms have developed assuming the presence of ranging sensor. Examples of ranging sensor are like sonar sensor, laser sensor, optical flow sensor and depth sensor. Unfortunately, all these sensors are suffering from various issues that will be affected the outcome of the robot as required. This type of sensors is replaced by machine vision which allows fully understanding of the environment. The camera operates on a wide range of lighting condition, are lightweight and low-powered enough to allow deployment in a variety of platforms. The challenge of image processing is the translation speed and fortunately deep learning has given great promises in image-based problem.

Deep learning provides appropriate features by manipulating the pixel information and never rely on camera parameters, geometry and camera calibration. The advanced technology in GPU promises deep learning can work well at frame rate, and provide the fast enough for accurate robot control.

2.4.1 Static

Lei T, Shaohua L, Ming L (2016) were provided two types of environment, if the robot should have specific training and provide the complicated junction environment in order the agent to make a decision by avoiding the obstacle (wall). Here the writer used soft-max classifier for decision making.

Valeri K. and Jianli Y. (2011) presented the obstacle network like in the Figure 13.

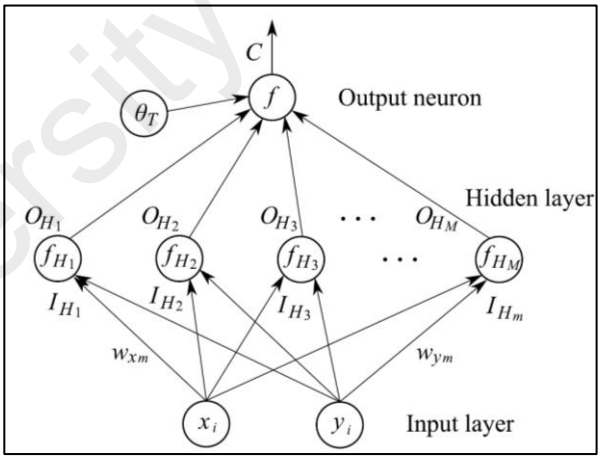

Figure 13: Obstacle description by neural network

Input of the network is the coordinates of point of the path denoted by two-dimensional environment x_i and y_i. Meanwhile the output layer is a repulsive penalty function f which is has a role of repulsive potential. Input of the

hidden layer is the weight and has the role of inducing local field of the neuron function and the output of the hidden layer is the number of vertices of the obstacles.

The writers present three types of obstacles; polygonal, circular and elliptical. All types of obstacle shapes have their own equation of induced local field.

Another important element in defining the obstacle is a local minimum where it will contribute to the inefficient of potential field methods. The way to defining better local minima is by:

1. Definition of potential function

   Specify a function with no or less local minima.
2. Design of search algorithm

   Include appropriate techniques for escaping local minima.

In order to define the local minima problem in the simplest way, the writers addressed them in three different ways:

1. Coordinate of start and destination is set respectively in Figure 14 (a). The polygonal obstacles are scanned to find the concavity like in Figure 14 (b). During the process of detecting, every two others vertices of a given obstacle are linked by straight line. A concavity happened when the line is outside the obstacle.
2. If one of goal and starting point is located inside a concavity, a temporary goal or start point is set by lying outside the concavity like

in Figure 14 (b). New straight line is created on the temporary point. This temporary point is set at the nearest possible to the original point.

3. After completing current path-planning task, the original path is retained so that the next planning could be planned correctly.

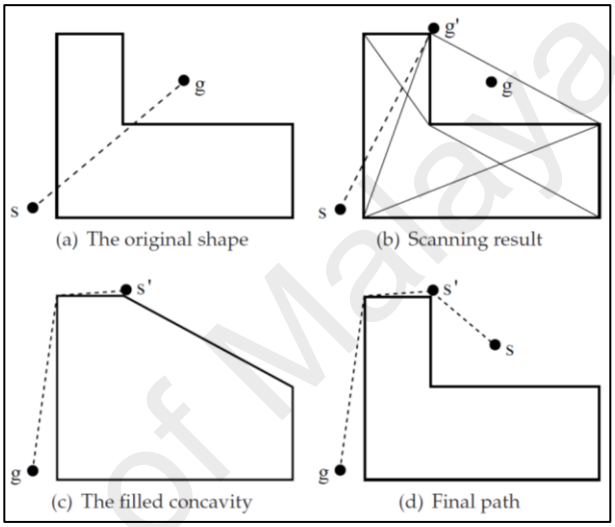

Figure 14: Local Minima Solution

Several steps concluded by Yang L., Shujuan Y., Yuroang L., Yuling J. (2016) in global rolling prediction algorithm for collision avoidance.

1. Parameters initialization

   The parameter needs to initialize like starting point, destination, robot step distance, rolling window radius.

   This global path is planned by the CNN.

2. The robot moves forward according to the window in the global path, update the current environment information in current window do check whether it reaches to destination or not. The algorithm will stop when it reaches destination.

3. The collision detection is running in the rolling window. If there is no collision then the robot moves forward a step. Process at Point 2 is repeated.
4. According to the collision avoiding strategy, the global path is planned, then the robot moves along this path and process at Point 2 is repeated.

For obstacle avoidance for flying object like quadrotor is presented by Punarjay C., Klaas K., Tom R., Stijn W., Tinne T., and Luc V. E. (2017). The writers use the depth maps which converted to 2.5D by using values from the centre in horizontal and vertical direction. For this kind of problem, the input of the quadrotor CNN is depth value di and the current angle of it. These values will be input to avoid behavior algorithm in order to find the best angular velocity where it is beneficial later to avoid behavior formula to allow the quadrotor reach the destination point.

Yang L., Shujuan Y., Yuroang L., Yuling J. (2016) proposed the solution on how to determine the nonviolent and the reasonable shortest reference path beginning from the present position of biomimetic robot which based on machine vision. The machine vision information contains the movement of robot speed and the movement direction of the obstacle. This local dynamic path planning strategy provides the local path planning which each of them must be having a reasonable length to allow the biomimetic robot to avoid the obstacle and planning the path which in unplaning area.

The detection of the path planning mentioned by the writer is to determine either the given window in the image contains the specific path part or not which contributing to the accuracy with the global reference path. A window contains partial path when there is the portion of the path part in it at least a particular length, which relative to the total length of the part.

Oliver S. (2018) created a robot purposely for obstacle avoidance. The robot receives different bytes which represent different direction of the robot. The table below shows the bytes receive from the controller of the robot for a different direction.

Instruction	Bytes sent
Forward	00100010 01111111 00000000
Forward-right	00100010 01111111 01100000
Forward-left	00100010 01111111 10100000
Backward	00100010 10000000 00000000
Backward-right	00100010 10000000 01100000
Backward-left	00100010 10000000 10100000
Rotate right	00100010 00000000 01111111
Rotate left	00100010 00000000 10000000

Table 1: The instructions that are sent when controlling the robotic car.

The data that used for training is collected by using robotic car and the testing data is being done in several places like in the office, house and lab. The testing data done in office shows that the robot be able to avoid the obstacle like wall, chair legs, bookshelf, bar stool, couch, bin, conference table, backpack and fire extinguisher. To have accurate on turning the robotic car from hitting the obstacle, the distance of the robotic car and the obstacle must be the same distance on every image data train.

Joseph D. and Brandley H. (2018) has two inputs to the network, the distance and the angle of the robot. The obstacle is discretized into wedges and the nearest obstacle can be detected by the robot by using laser scanner. The existence of the obstacle is represented by 1 while 0 shows no obstacle near by the robot.

2.5 Conclusion

Lei T, Shaohua L, Ming L (2016) evaluate the output in two aspects. The consistency of the robot depends on the generated commands with the reference of human operated command. Next one is how the similarity between the agent (robot) exploration trajectory and human exploration trajectory. The result is by plotting the angular velocity over the time by setting the linear velocity to constant. In this case, the writers confirm their agent (robot) is able to decide and make the decision likely to human.

The input of the CNN used by Keith S., and Wallace L. (2016) is RGB image where the issue of salience in the image. This might be not a problem when it is in constrained environment, but it may take into consideration when the environment is low hanging obstacles.

The challenge of deploy supervised learning is collecting labelled training data where it is time consuming, error prone and not in large scale action spaces.

CNN application is widely used either on land or water. Joel, O. G., Lucas, T. G., Amanda, C. D., Breno, Z., Paulo, D.-J., & Silvia, S. C. (2016) had implemented the

Autonomous Underwater Vehicle and the research is about the collision free when the vehicle is under the water circumstance.

CHAPTER 3

METHODOLOGY

3.1 Project Methodology

This chapter discusses on the implementation method of the project. The project is done by simulation using MATLAB software. The neural network used in this project is Convolutional Neural Network with supervised learning by using labelling data training. Data training is divided into eight different folders representing the directions of the robot that should be taken. The network is given three different environments or maps with three stages of difficulties to evaluate the accuracy and effectiveness of the network.

3.2 Project Block Diagram

Diagram 4 shows how the project is flow. The robot is located at the initial position and it needs to check its surrounding environment before do the movement. After safe position is determine, the system captured the current position and fed it to the CNN to make the prediction. These all steps are repeated until the robot reach the destination position. But, if the robot hit the obstacle, CNN require to retrain due to the looping prediction by the CNN.

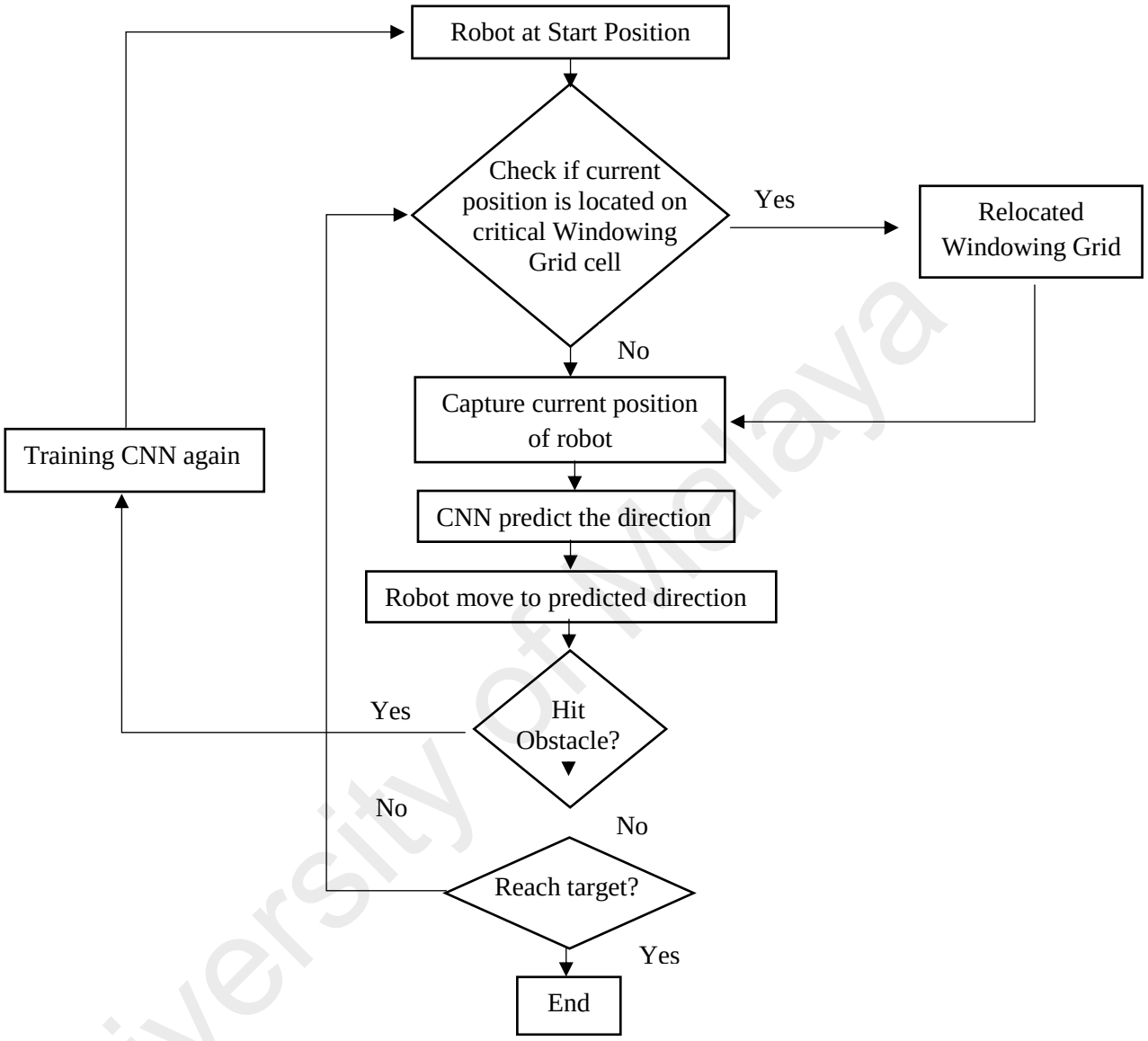

Diagram 4: Project Block Diagram

3.3 Convolutional Neural Network Architecture

The neural network consists of Convolutional layer, ReLU layer, Max-Pooling layer, Fully-connected layer, Softmax layer and Classification layer. In this project, CNN works in time series in order to predict the direction that the robot should take to reach the target position.

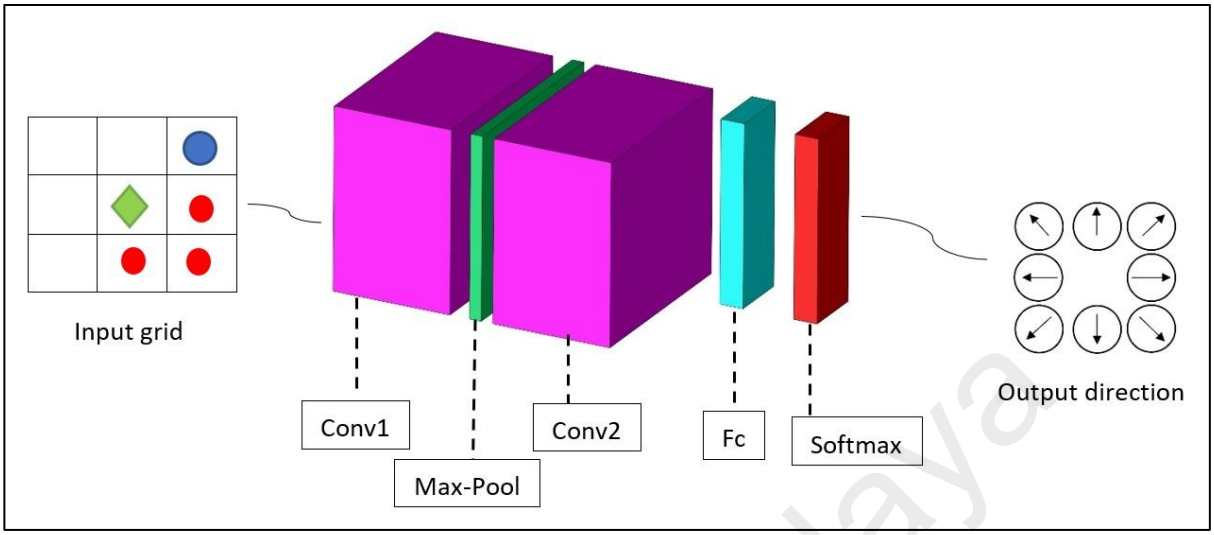

Figure 15: The architecture of the Convolutional Neural Network

This network uses label image for training. Each of the images is representing the eight directions which robot to take on different kinds of position. The training data size is 66×88 in RGB image type. The training image itself has different types, where it contains only target and start position, and free space or second type target, start position, free space and static obstacles.

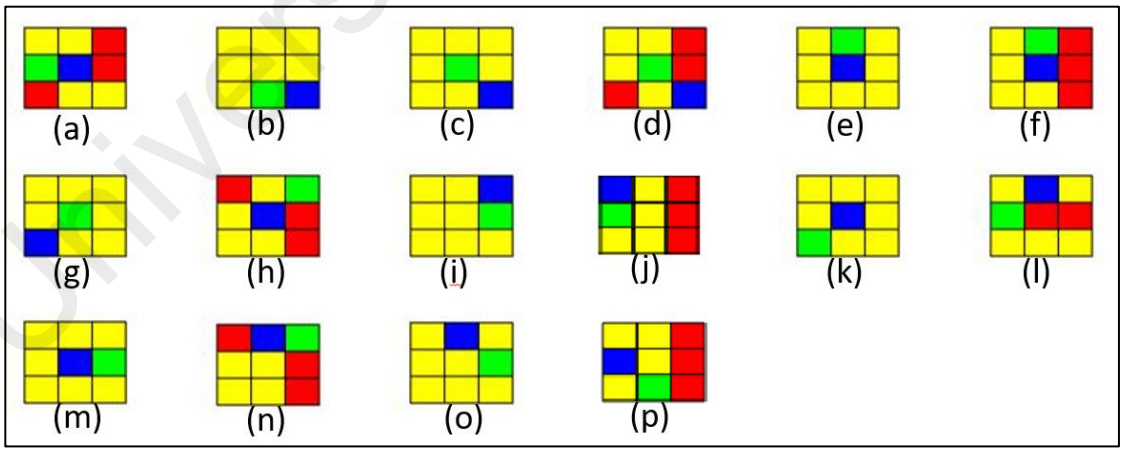

Figure 16: 3by3 matrix representing the current position

From the above figure, the red cell is represented obstacle, blue is the start or current position and green is the temporary target. From the Figure 16, (a) and (b) is

representing DOWN direction, (c) and (d) is representing DOWN-LEFT direction, (e) and (f) representing LEFT direction, (g) and (h) is representing LEFT-UP direction, (i) and (j) is representing RIGHT direction, (k) and (l) is representing RIGHT-DOWN direction, (m) and (n) is representing UP direction and (o) and (p) is representing UP-RIGHT direction.

Preparing the data is very crucial because it requires complete information, especially on how to locate the obstacle. Mostly the training data is required from the simulation map. The training data is made from 3×3 matrix which representing from eight possible directions of the robot that could take.

The testing data to the network is 3×3 matrix of current position, where the free space is representing yellow cell, red is representing obstacle, blue is representing current position and green is representing the temporary target. In the other hand, the network is using supervised learning for the learning process.

There will be a sequence of images created in order to allow the robot move from its initial position to the targeted position in the grid map. From that image the network will make the prediction for the best next move it should take.

For the Convolutional layer, the command works in MATLAB is *convolution2dLayer(filterSize,numFilters,Name,Value).* In this project, the filter size of first Convolutional layer 3 and the number of filters is 16 while the second Convolutional layer is filter size is the same, and the number of filters is 32.

Each of the Convolutional layer will proceed with Batch-normalization Layer and ReLU Layer.

For Max-Pooling layer command in the MATLAB software it is *maxPooling2dLayer(3,'Stride',2,'Padding','same')* where the stride is 3 and the padding value is 2.

The network then proceeds with fully-connected layer, Softmax layer and end with classification layer.

The network is being trained with the training option like in the following command:

```
29 -    opts = trainingOptions('adam', ...
30          'InitialLearnRate',3e-4, ...
31          'SquaredGradientDecayFactor',0.99, ...
32          'MaxEpochs',80, ...
33          'MiniBatchSize',64, ...
34          'Plots','training-progress')
```

Figure 17: Training option of the network

In the figure above, the network is being train using *adam* solver with the learning rate is 3e-4, and the maximum epochs is 80.

3.4 Map in MATLAB

There are three types of map for the CNN to work on. Each map has different positions for start and target. At the same time, the obstacle's location and their structured arrangement are from easy to difficult level. The start position is labelled as blue circle and the destination is in the shape of green diamond. Meanwhile the obstacle is a red circle. The obstacle involve in these maps is a static obstacle where they locate in horizontal, vertical and diagonal position. The map is built by grid with 10×10 matrix grid. The robot has possibility to take only one direction from eight different directions at one time. The available direction allows the robot to move is

UP, UP-RIGHT, RIGHT, RIGHT-DOWN, DOWN, DOWN-LEFT, LEFT, LEFT-UP.

3.4.1 Map 1

For Map 1, this is where the easy stage of the testing map for CNN. The start position is at (1,1) position and the target is at (10,10) position in the grid. The obstacle is in position of horizontal and vertical.

The position of obstacles are at (1,3), (2,3), (3,3), (2,5), (2,6), (2,7), (6,1), (6,2), (6,3), (6,4), (6,5), (3,9), (4,9), (5,9), (6,9), (9,6), (9,7), (9,8), (9,9) and (9,10).

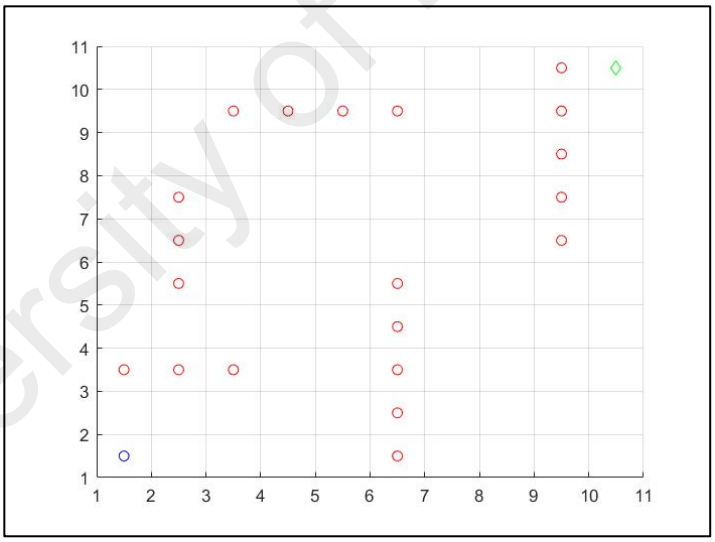

Figure 18: The environment of Map 1

The robot should avoid all the obstacles. If this case is happening, the robot should start at its initial position which is located at (1,1).

3.4.2 Map 2

Figure 19 is the second testing map for the CNN. The difficulty level is medium. The robot should be able to move besides the obstacle for 3 rows at first. It should avoid from moving down of the maps when it's about to reach the center of the map. Turning should be made at the right-bottom corner of the map before it reaches to the destination.

Initial position given to the robot is at (1,10) and the targeted position is at (10,1).

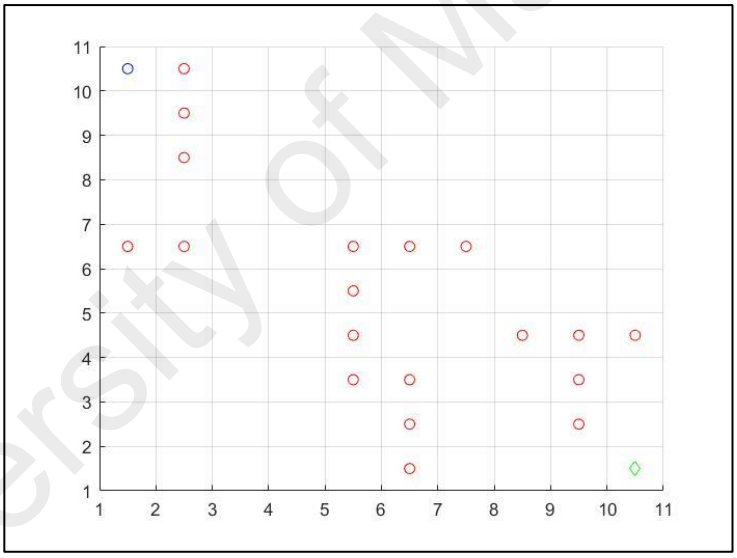

Figure 19: Environment of Map 2

The obstacle position in Map 2 is a bit different compared with Map 1. For Map 2, the arrangement of the obstacle is a combination of horizontal and vertical. First obstacle the robot should move down for three rows is located at (2,8), (2,9) and (2,10). This position is to test if the robot understands to move straight without turning in any direction.

Obstacle 2 is at position (1,6) and (2,6) while Obstacle 3 is a combination of horizontal and vertical position at (6,1), (6,2), (6,3), (5,3), (5,4), (5,5), (5,6), (6,6), and (6,7). In between of Obstacle 2 and 3, there are 2 empty grid space. The purpose of this empty is to test the robot if it can do correct prediction of the next best move when there's no existing of obstacle near to it.

End of Obstacle 3, the turning should be made in order to avoid the robot from hitting the Obstacle 4 where it is located at position (8,4), (9,4), (10,4), (9,2), and (9,3). Another turning from the robot should be made to allow it to reach the destination at the bottom of Obstacle 4.

3.4.3 Map 3

This is the last testing map where the difficulty level is high compared to Map 1 and Map 2. The start position is at (10,10) and the targeted position is at (3,4). Obstacle now is more complex compared to two initial maps. Turning and moving along the obstacle is more. Now the target position is in between of the obstacle and require turning in order to reach the destination point.

Obstacle 1 is where it's look like C-shape and the position is at (8,4), (7,4), (6,4), (6,5), (6,6), (6,7), (6,8), (7,8), (7,9), (8,9), (9,9) and (9,10). The small obstacle surrounded by Obstacle 1 is Obstacle 2 which locates at the point (8,6), (9,6), (10,6) and (10,7). The existence of Obstacle 2 there is to test the robot if it able to move in diagonal direction. The difficult part continues when it starts to leave the C-shape of Obstacle 1 and ready to turn towards Obstacle 3.

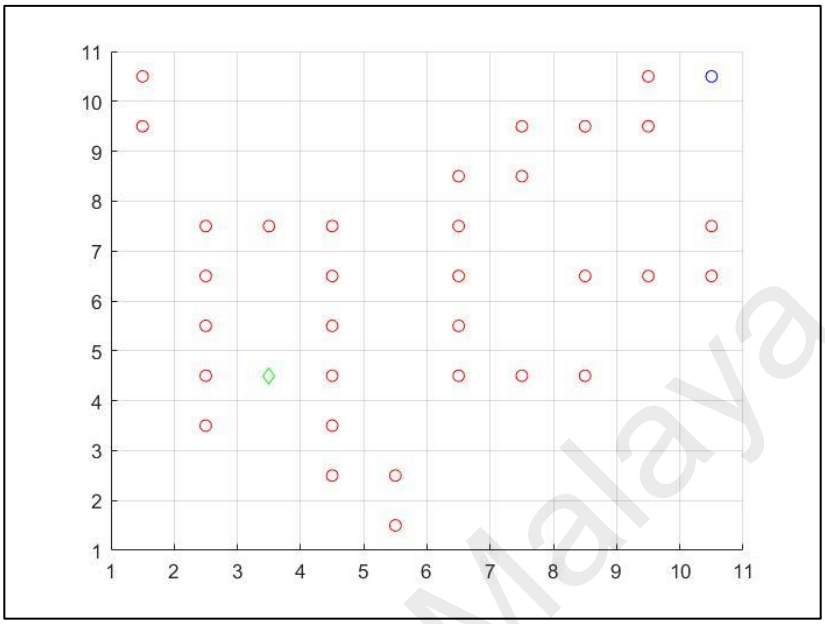

Figure 20: Environment of Map 3

The robot should again move along the narrow obstacle which in between Obstacle 1 and Obstacle 3. Second narrow path is between Obstacle 3 and the left side of grid frame.

Obstacle 3 is located at point (2,3), (2,4), (2,5), (2,6), (2,7), (3,7), (4,7), (4,6), (4,5), (4,4), (4,3), (4,2), (5,2) and (5,1).

Challenging of Obstacle 3 is when the robot should move along the obstacle and the grid frame. The 5 movement of down should be made before it turns to reach the target position where it is surrounded by the obstacles.

3.5 Capturing Current Position by Windowing Grid

The network is unable to train 10×10 grid with higher accuracy, so capturing current position in 3×3 grid is required. The 3×3 grid consists of current position, temporary target, obstacle (if available) and the free space. This 3×3 grid which capturing the current position of the robot is called Windowing Grid.

This Windowing Grid image will be fed into the network to make a prediction on the next best move to approach the target position.

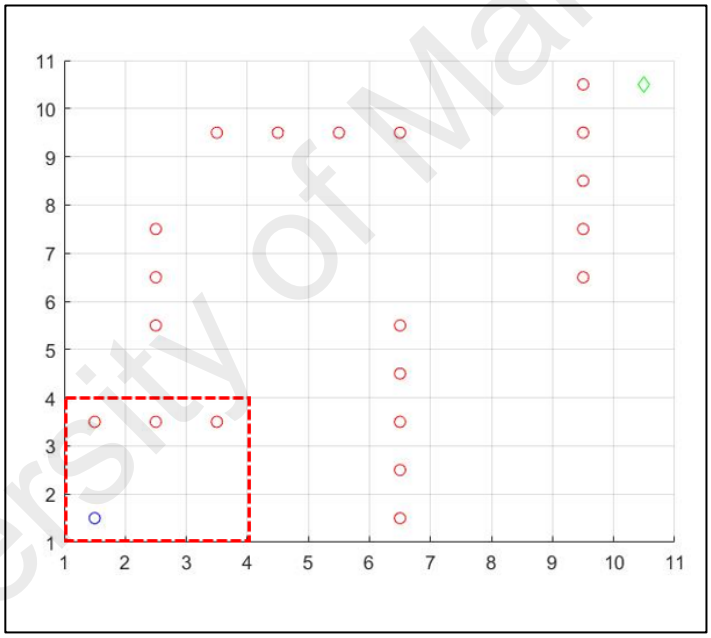

Figure 21: Windowing Grid is the red box with the dotted line. Cover the area 3by3 grid at start position

The current position of the robot should update to ensure it was always in cell of the Windowing Grid after it do the movement. This is to ensure the current position of the robot is always lies inside the Windowing Grid coverage. If the robot fails to update its position, the image created to fed the network is incomplete and may contribute to wrong prediction.

W11	W12	W13
W21	W22	W23
W31	W32	W33

Table 2: Cell variables for the Windowing Grid

Table 2 shows the cell of Windowing Grid which overall cell contains is 9 cells. Every cell has their own position. W11 represent cell inside the Windowing Grid at row number 1 and column number 1. While W12 represents cell where it is located at row number 1 and column number 2.

3.5.1 Windowing Grid Along Tested Frame

The Windowing Grid will require to move when it is in several conditions. Figure 16 shows the Widowing Grid constantly move along the frame of the grid.

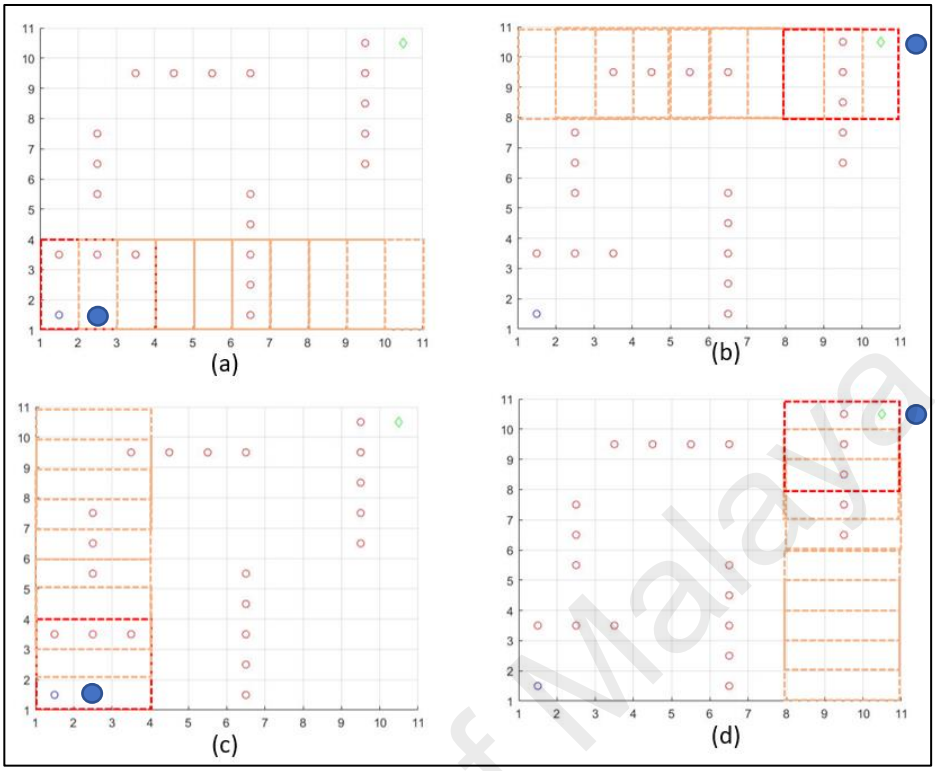

Figure 22: Windowing Grid moves along the frame

The Windowing Grid constantly moves to the right like shown in Figure 22 (a) when the temporary target is located at the right cell of Windowing Grid which is at w13, w23 and w33. The windowing Grid will move to the left like in Figure 22 (b) when the temporary target is located at the left side of Windowing Grid which is at w11, w21 and w31.

Figure 22 (c) shows the Windowing move upward when the temporary target is located at cell w11, w12 and w13. The Windowing Grid move downward like in the Figure 22 (d) when the temporary target's position is located at bottom side of Windowing Grid which is at w31, w32 and w33.

3.5.2 Windowing Grid in the middle of map

There are different situations should be set on the Windowing Grid when it is located in the middle of maps or where it's not touching to the map's frame. In the beginning of the chapter also mentioned that Windowing Grid is resulting from 3×3 grid, which will be later used to feed the network for prediction on the next best move.

Table 3 shows the Windowing Grid which each cell has their own variable. The shaded cell is where the effective cell which requires attention to move the Windowing Grid when the current position is lying on them.

w11	w12	w13
w21	w22	w23
w31	w32	w33

Table 3: Shaded cell is the effective cell require attention to relocate the Windowing Grid

There are several situations when the current position is located on the shaded cell like in the Table 3. There are when:

i. Current position lies at w11

When the current position lies on cell w11 of Windowing Grid, there are three conditions where the Windowing Grid should relocate its position.

W11	W12	W13
W21	W22	W23
W31	W32	W33

Table 4: Windowing Grid with shaded cell at w11

Case 1: When the Temporary Target is located at the left side or bottom-left of the current position

Given the current position of the robot in the Figure 23 (a) is at (6,3) and the temporary target is located at (5,3), one column lower from the current position. The current position of the robot is located at cell w11 of Windowing Grid, so in order to capture appropriate input data test for CNN to predict, the Windowing Grid should move to the left by one row.

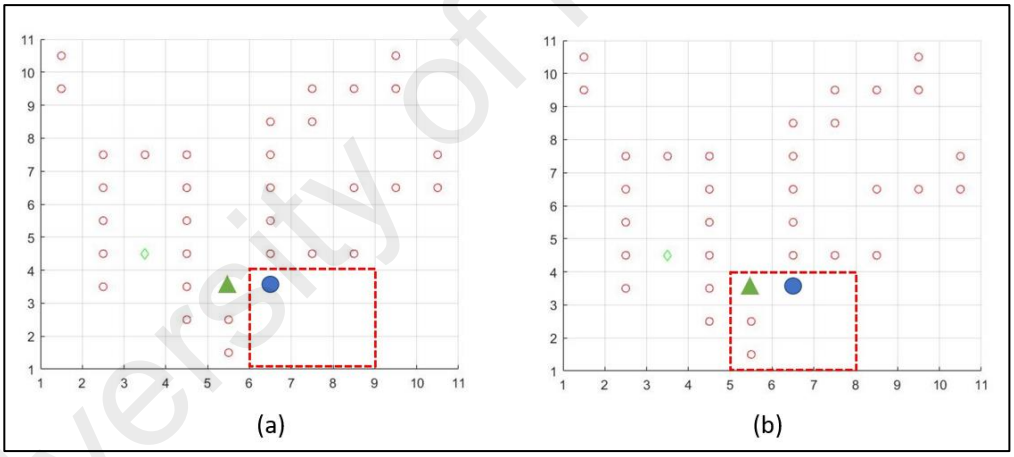

Figure 23: (a) the temporary target is located beside current position, (b) Windowing Grid position after shift to left

Shifting down of Windowing Grid also applied to the situation when the temporary target is located at the bottom-left from the current position like shown in Figure 23 (a).

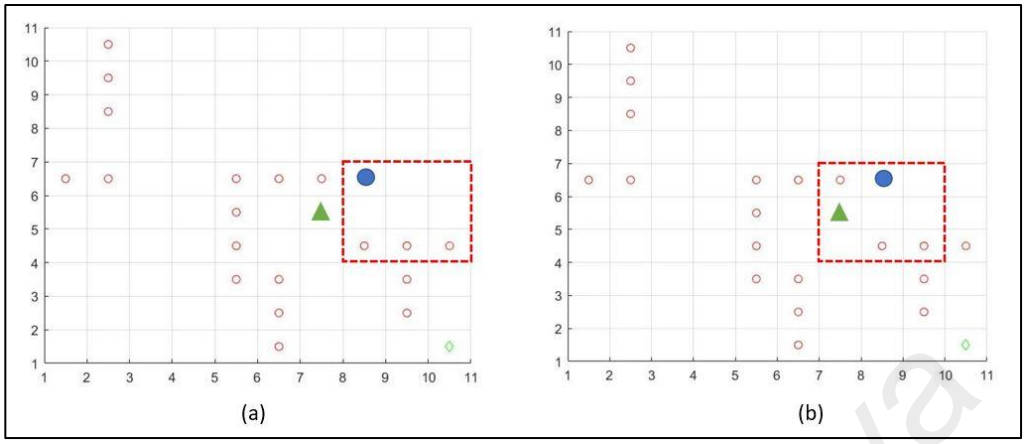

Figure 24: When the temporary target is located at bottom-left from the current position and (b) is the resulting position after shifting the Windowing Grid to downward

Case 2: Temporary target is located at top or top-right location from the current position

In Figure 25 (a) and (b), both of them show the temporary target is locate at top or top-right from the current position of the robot. For Figure 25 (a), the current position is located at (5,4) and the temporary target is located at (5,5) which one row upper from the current position of the robot. Meanwhile in Figure 25 (b), the current position of the robot is located at (5,5) and the temporary target is located at (6,7) which advance in both column and row by one.

In both situations, the Windowing Grid should move upward by one row in order to capture informative data test to feed the network.

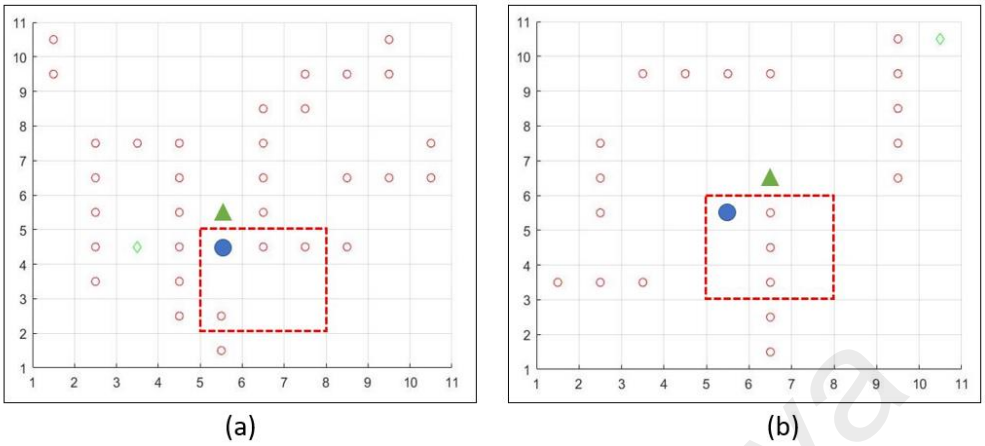

Figure 25: Situation of where the current position lies on w11 and the temporary target is located at top or top-right position

Figure 26 shows each of them after the Windowing Grid shifted to upward by one row.

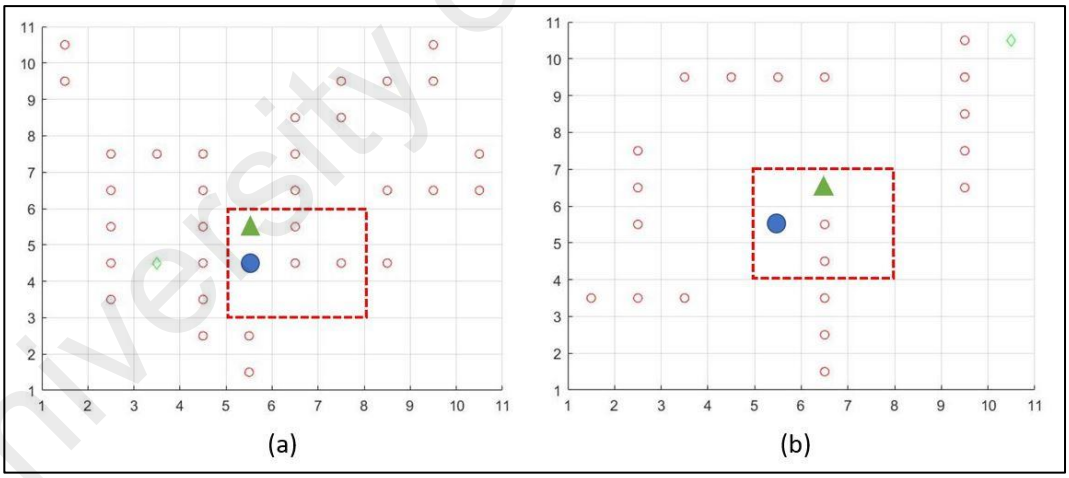

Figure 26: the position of Windowing Grid after shifted to upward for both cases

Case 3: Temporary target is located at the top-left from current position.

Referred to Figure 27 (a), the temporary target is located at (5,4) where it is lower by one column and upper by one row from the current position of the robot which

located at (6,3). In this situation, the Windowing Grid needs to move diagonally up and shifted to left by one column in order to capture the complete data test.

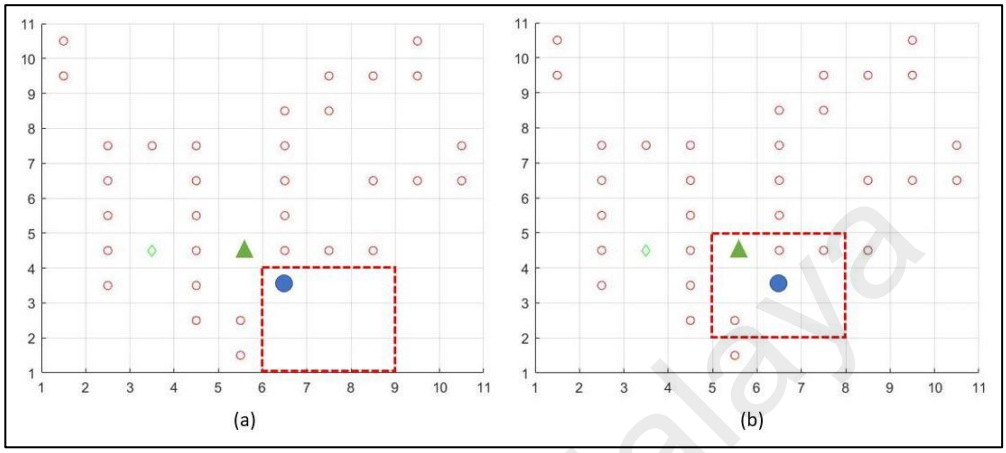

Figure 27: (a) The position of where the Windowing Grid should move diagonal, (b) final position of Windowing Grid after shifted

ii. Current position lies on w12

There are only three states of temporary target location when the current position of the robot lies on cell w12 if Windowing Grid.

W11	W12	W13
W21	W22	W23
W31	W32	W33

Table 5: Windowing Grid with shaded cell at w12

There are in Figure 28 (a) where the temporary target is located at top-left, or Figure 28 (b) where it is just at the top, or like in Figure 28 (c) where it's located at top-right from the current position.

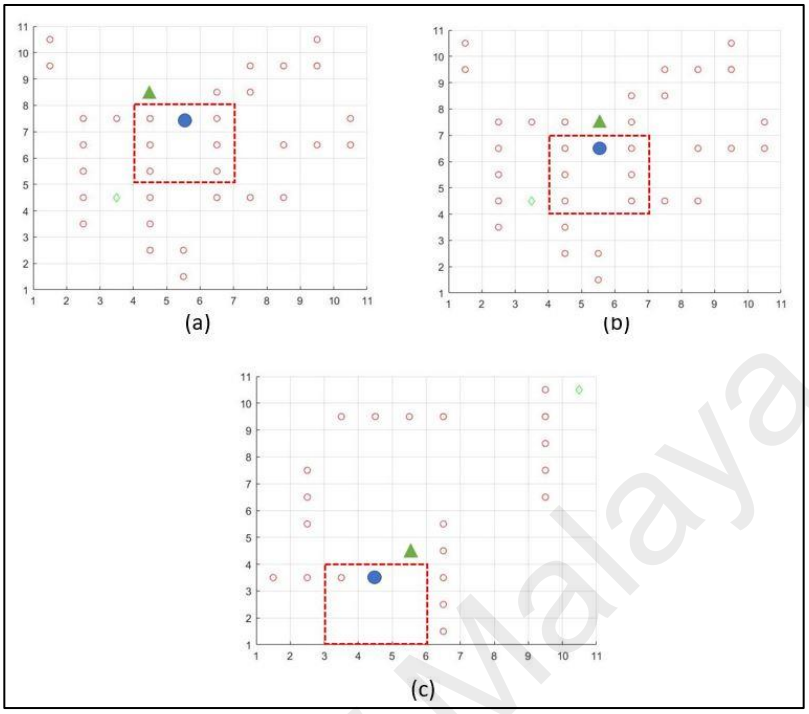

Figure 28: Three states when the current position is lies on w12

Each of these states show that the Windowing Grid should move upward by one row in order to get the correct data test. The result of each state stated in Figure 29.

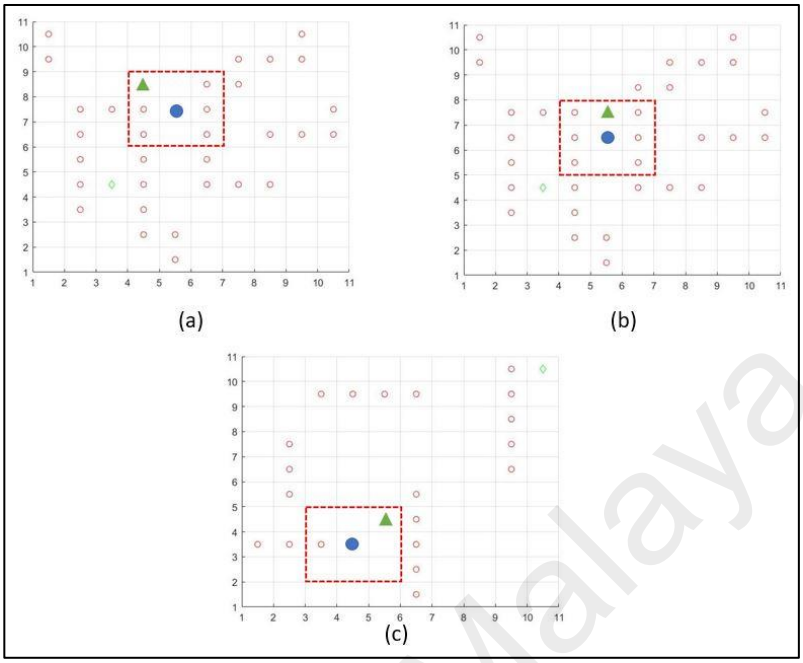

Figure 29: The position of Windowing Grid after shifted to up for each state

iii. Current position lies on w13

There are three states where the Windowing Grid should shift its position. It's when the current position lies on w13 of the Windowing Grid cell.

W11	W12	W13
W21	W22	W23
W31	W32	W33

Table 6: Windowing Grid with shaded cell at w13

All these three states described as follows:

Case 1: Temporary target is located at top-left and top of current position

Referred to Figure 30 (a), the current position of the robot is located at position (6,3) while the temporary target is located at (5,4). Here, the

temporary target locates lower by one column and upper in one row from the current position. While in Figure 30 (b) the temporary target is just in the position one row upper from the temporary target.

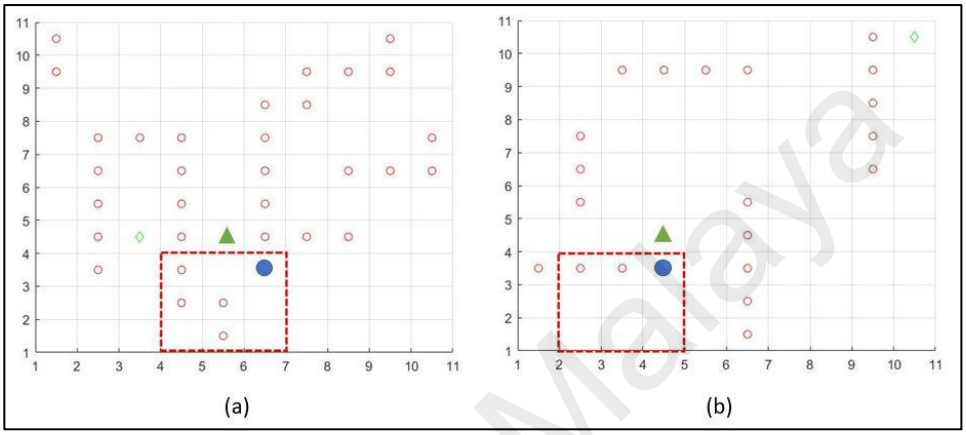

Figure 30: (a) is where the temporary target is at top-left position, (b) is where the temporary target is located at top of current position

In these two locations of temporary target (the current position lies on cell w13 of Windowing Grid) thus, the Windowing Grid should move upward by one row in order to capture necessary data test to feed into CNN.

Figure 31 (a) and (b) show the position of Windowing Grid after it relocate by moving up by one row.

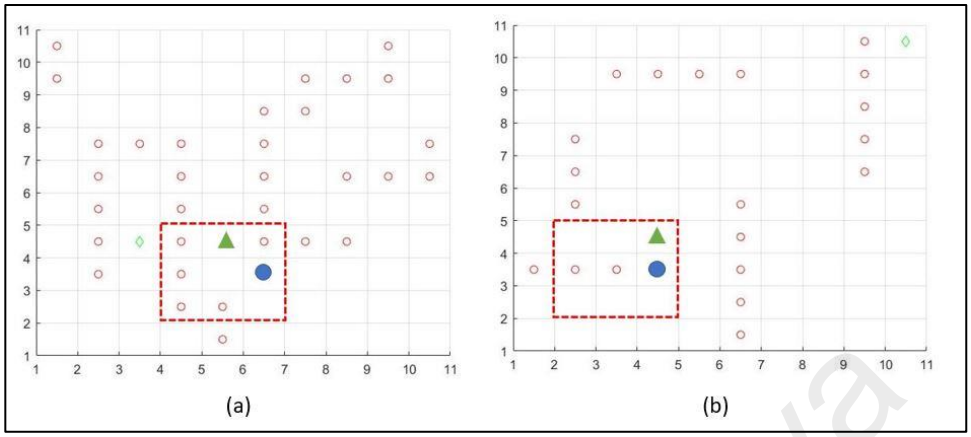

Figure 31: The resultant of Windowing Grid position after moving up by one row

Case 2: Temporary target is located at top-right of current position

In Figure 32 (a) the temporary target is located at (6,6) while the current position of robot is located at (5,5). In these positions the temporary target is located at top-right from the current position.

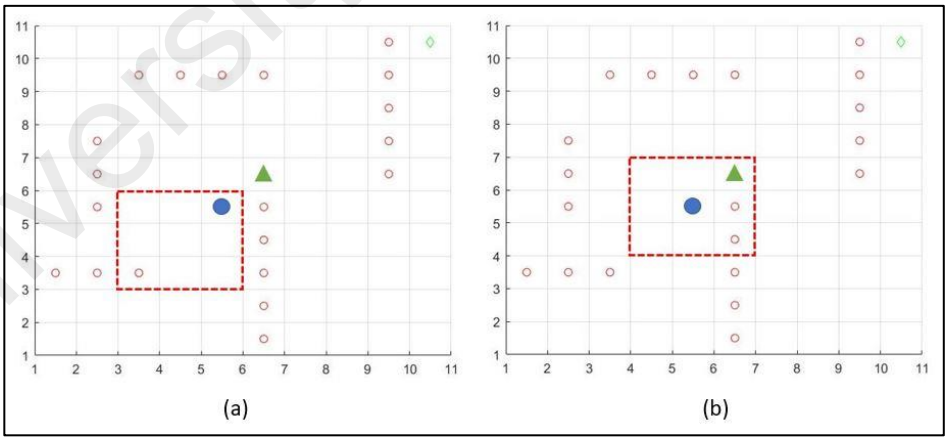

Figure 32: Temporary target is located at top-right position, (b) is the resultant of Windowing Grid after shifting

In order to collect informative input data test, the Windowing Grid should shift to the right by one column and move upward by one row. The final location of Windowing Grid of this problem is shown in Figure 32 (b).

Case 3: Temporary target is located at right and bottom-right of current position

There are two states where the Windowing Grid should be moving to the right when the current position lies on cell w13 of Windowing Grid. First state is when the temporary target is located at the right side of the current position, and second state is when the temporary target is located at the bottom-left of the current position. Each of the states can be found in Figure 33 (a) and (b).

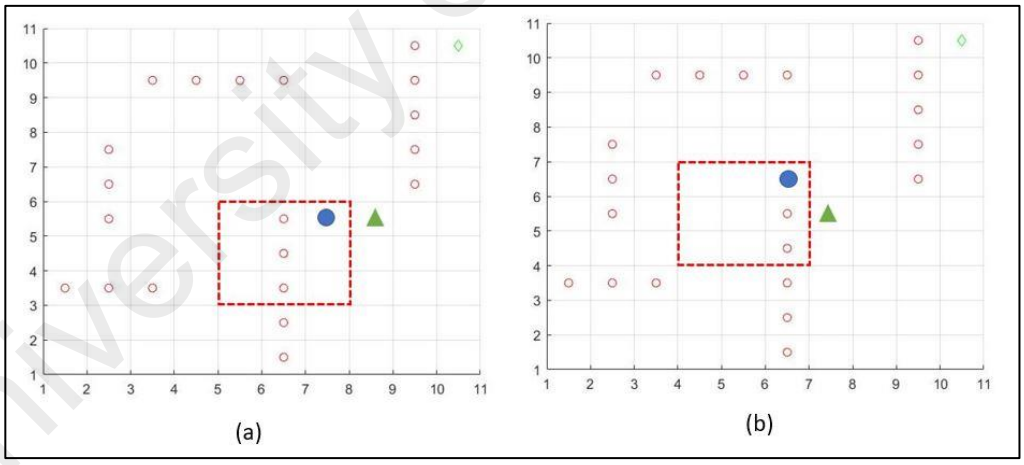

Figure 33: (a) Temporary target is located at right side, (b) temporary target is located at bottom-right

Figure 34 (a) shows the Windowing Grid move to the right when the temporary target is located at the right side of current position, while Figure 34 (b) shows the final position of Windowing Grid after it shift when the temporary target is located at bottom-right of the current position.

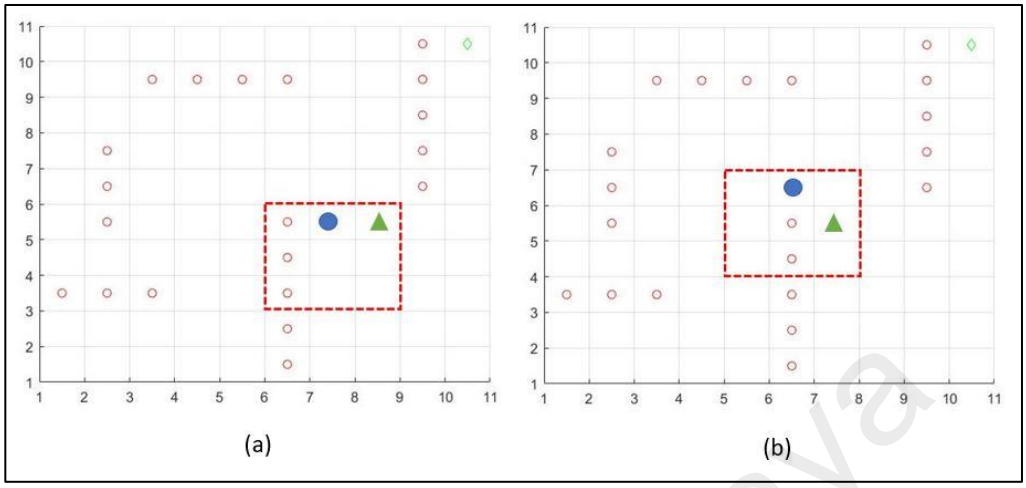

Figure 34: Windowing Grid move to right by one column for both conditions

iv. Current position lies at w21

In this state, only one movement taken by Windowing Grid on three conditions of where the temporary target is located. Table 7 shows the current position lies on cell w21 of Windowing Grid.

W11	W12	W13
W21	W22	W23
W31	W32	W33

Table 7: Shaded cell is where the current position is lies on Windowing Grid cell

From Figure 35 (a), the current position is located at position (6,3) and the temporary target is located at (5,4). Temporary target is said located at top-left from the current position of robot.

Figure 35 (b) shows the current position is located at (4,8) and the temporary target is located at (3,8) just one column lower than the current position. It is located at the left side of current position.

Figure 35 (c) is where the temporary target is located at (1,7) and current position is located at (2,8) where the temporary target is located in one row and one column lower from current position.

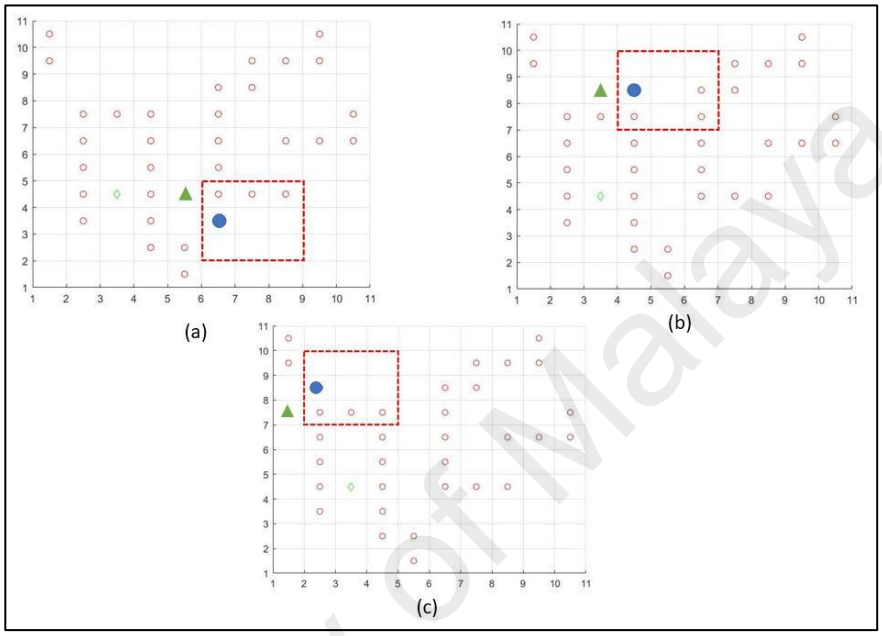

Figure 35: (a) temporary target is located at top-left from current position, (b) temporary target is located at left side from current position (c) temporary target is located at bottom-left from current position

Figure 36 shows the final position of Windowing Grid after it moves to the left by one column of each of the states mentioned in Figure 35.

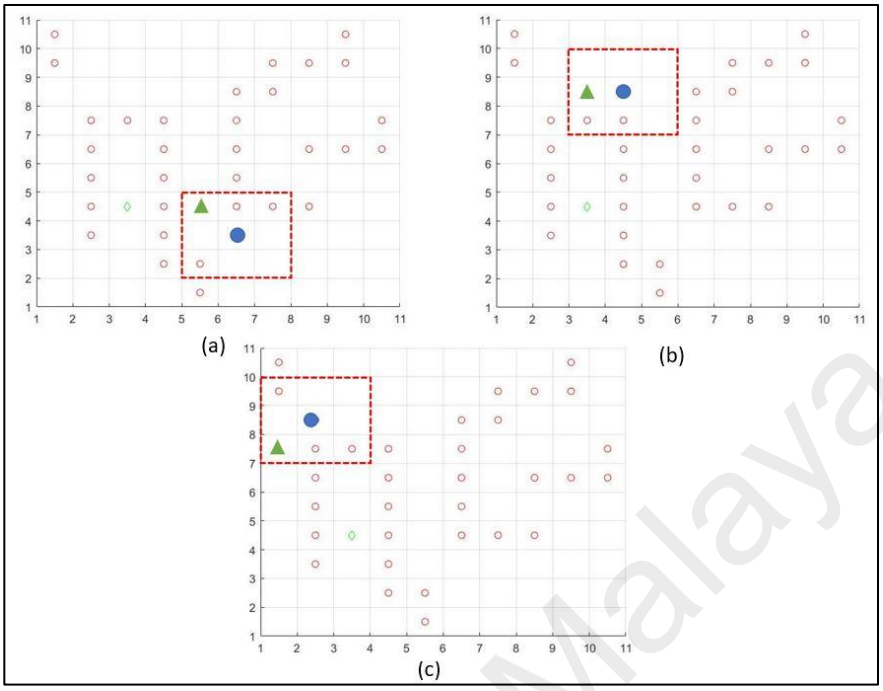

Figure 36: Final position of Windowing Grid after shifted to left by one column

v. Current position lies at w23

This is the condition when the current position of the robot lies on cell w23 of Windowing Grid like in Table 8.

W11	W12	W13
W21	W22	W23
W31	W32	W33

Table 8: The Windowing Grid with shaded cell at variable w23

There are only three positions of temporary target could be affected the position of Windowing Grid. There are mentioned in Figure 37 (a), (b) and (c).

In Figure 37 (a) is where the current position located at cell (3,2) and the temporary target at (4,3), which is upper by one column and one row from current position. While in Figure 37 (b), the temporary target is located on the right side of current position. And Figure 37 (c) is where the temporary target is located at bottom-right from the current position.

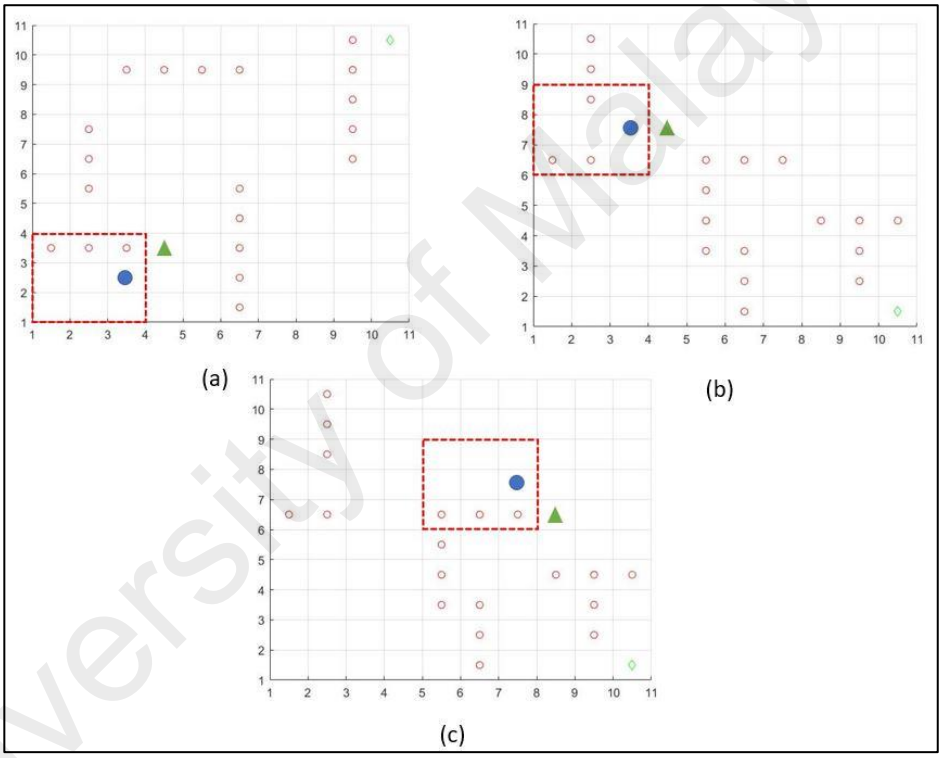

Figure 37: (a) Temporary target is located at top-right from current position, (b) temporary target located at right side of current position, (c) temporary target is located at bottom-right from current position

All of these conditions require the Windowing Grid to move one column to right in order to provide the necessary input data test to the network. The transformation of Windowing Grid for each condition available in Figure 38.

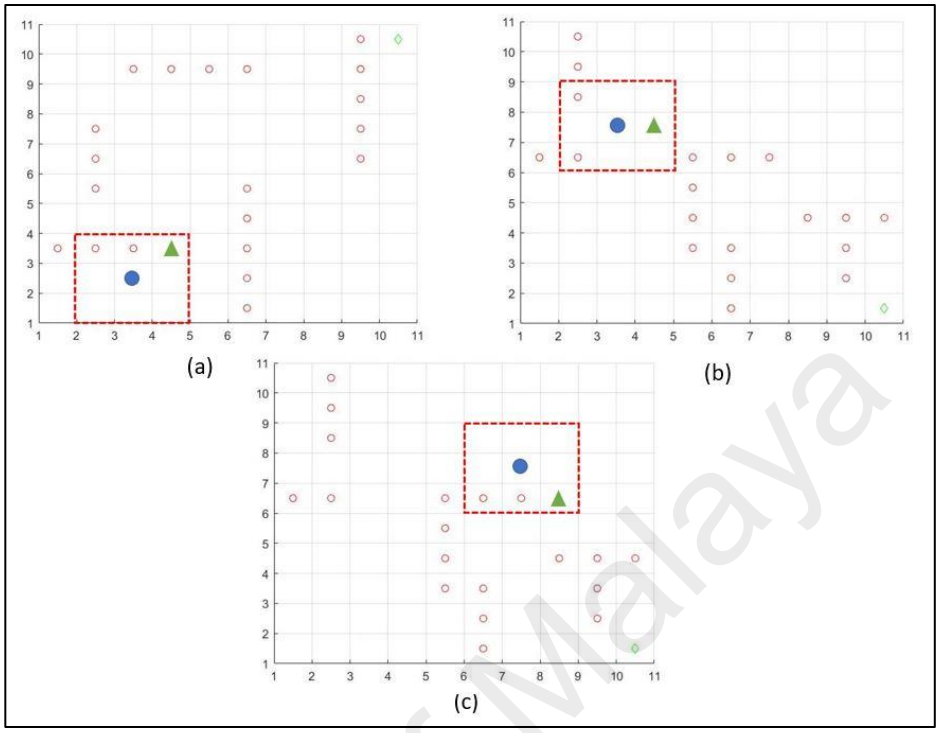

Figure 38: The state of Windowing Grid after transformation to the right

vi. Current position lies at w31

When the current position of robot lies on w31 cell of Windowing Grid, there are five situations where the Windowing Grid should face a movement.

W11	W12	W13
W21	W22	W23
W31	W32	W33

Table 9: Windowing Grid with shaded cell at variable w31

Case 1: Temporary target is located at up-left from the current position

In Figure 39, the current position of the robot is located at position (6,3) and the temporary target is located at (5,4).

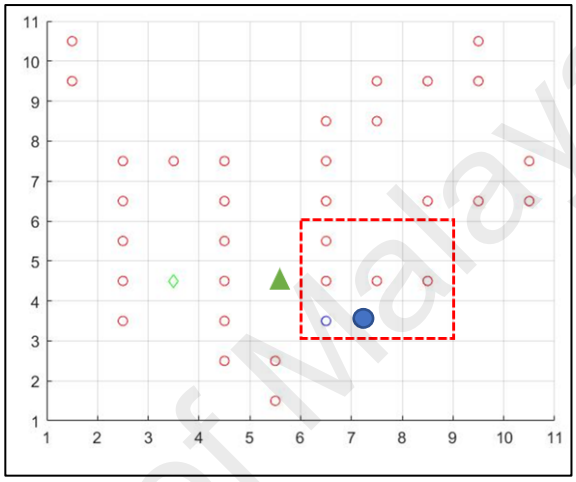

Figure 39: Current position lies on w31 and temporary target is located at up-left

In this situation the Windowing Grid should move left by one column and there are no changes on row position. The result after moving the window can be found in Figure 40.

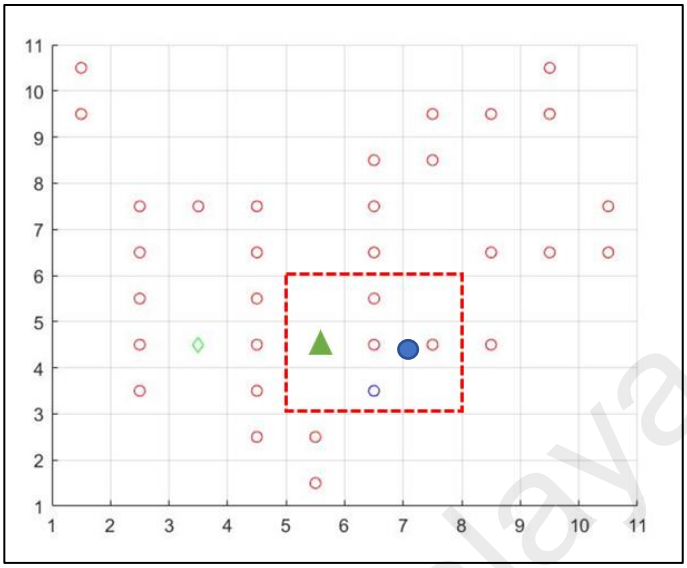

Figure 40: Windowing Grid move to left by one column

Case 2: Temporary target is located on the left side of current position

Referred to Figure 41, the current position is at (8,7) while the temporary target is located at (7,7) just beside the current position.

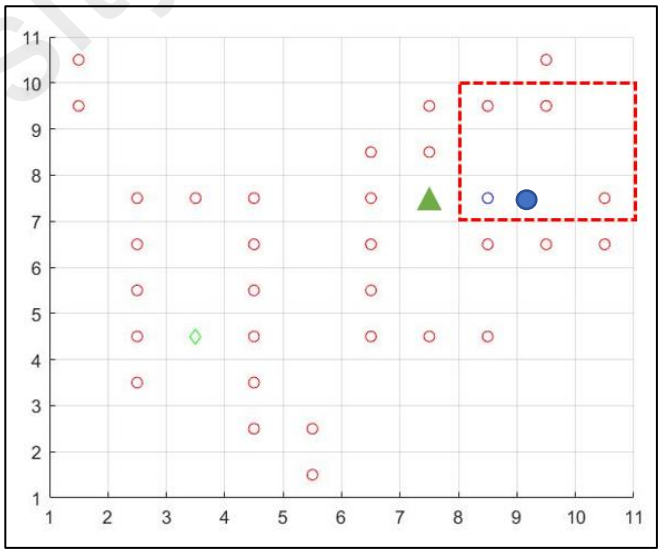

Figure 41: Temporary target is located on the left side of current position

In this case, the Windowing Grid should be moved left by one column and the result after moving is in Figure 42.

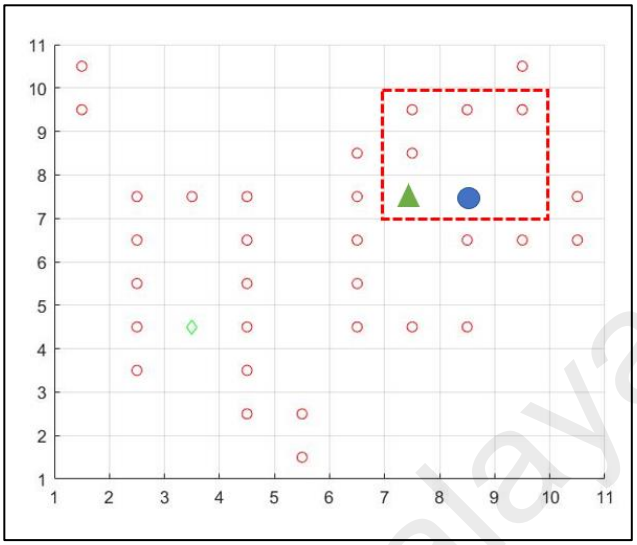

Figure 42: After Windowing Grid move to left by one column

Case 3: Temporary target is located at bottom-left of the current position

Now the current position is located at position (8,6) while the temporary target is located at point (7,5), bottom-left of the current position.

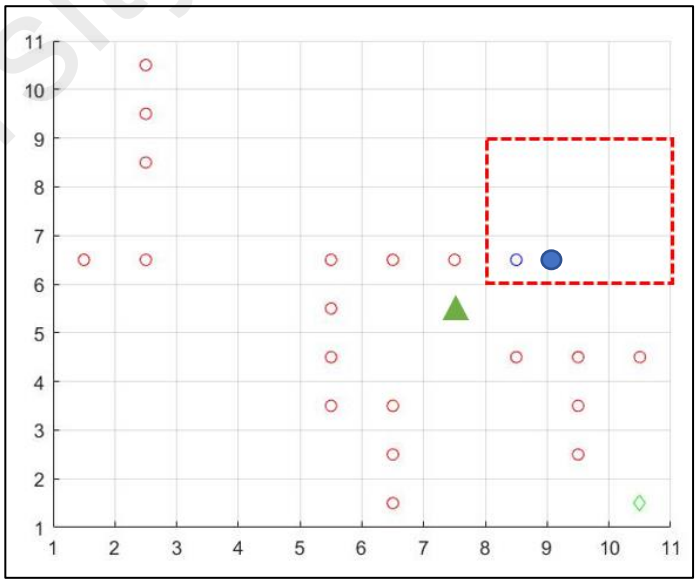

Figure 43: Temporary Target is located at bottom-left from current position

In this case, if the current position is located at cell w31 and the temporary target is located at bottom-left of the current position, the Windowing Grid should move down by one row and shift to the left by one column. The final position of this case is shown in Figure 44.

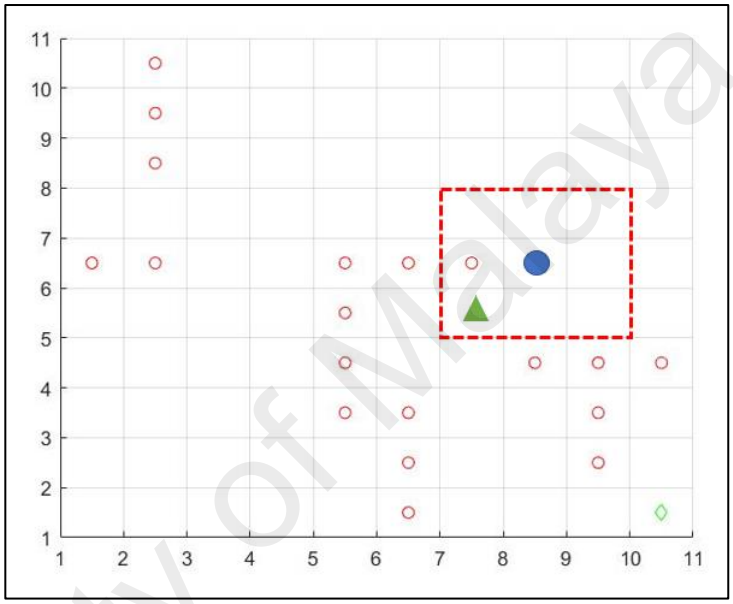

Figure 44: Windowing Grid moving down by one row and shift to left by one column

Case 4: Temporary target is located at bottom of current position

This case is like in Figure 45 where the current position is located at cell w31 of Windowing Grid and the temporary target is located at the bottom of it.

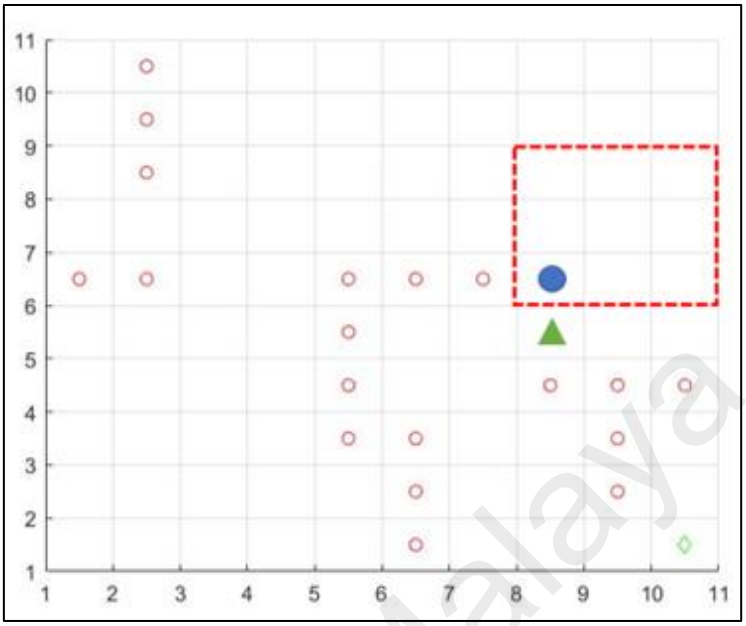

Figure 45: Current position is located at position (8,6) and temporary target is at (8,5), just below the current position location.

In this condition, the windowing Grid should move down by one row and the result like in the Figure 46.

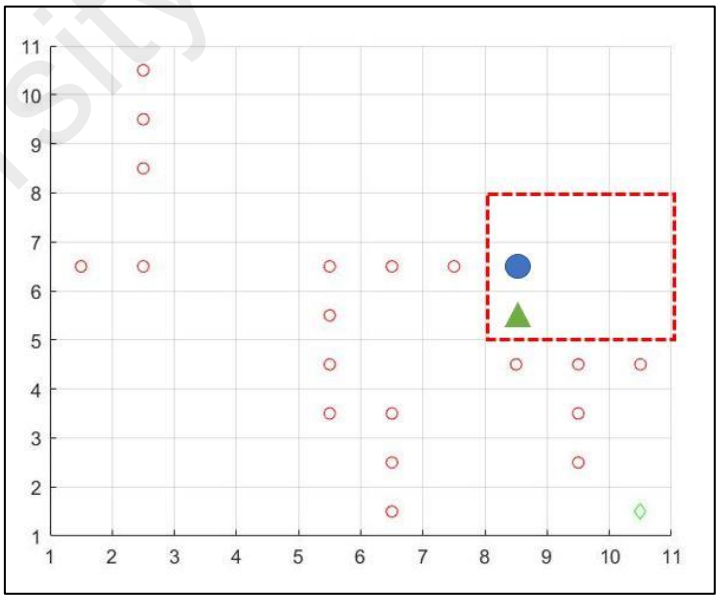

Figure 46: After the Windowing Grid move down by one row

Case 5: Temporary target located at the bottom-right of current position

Now the current position is located at position (8,6) and the temporary target is at (9,5) where it is one row lower than current position and with the extra one column.

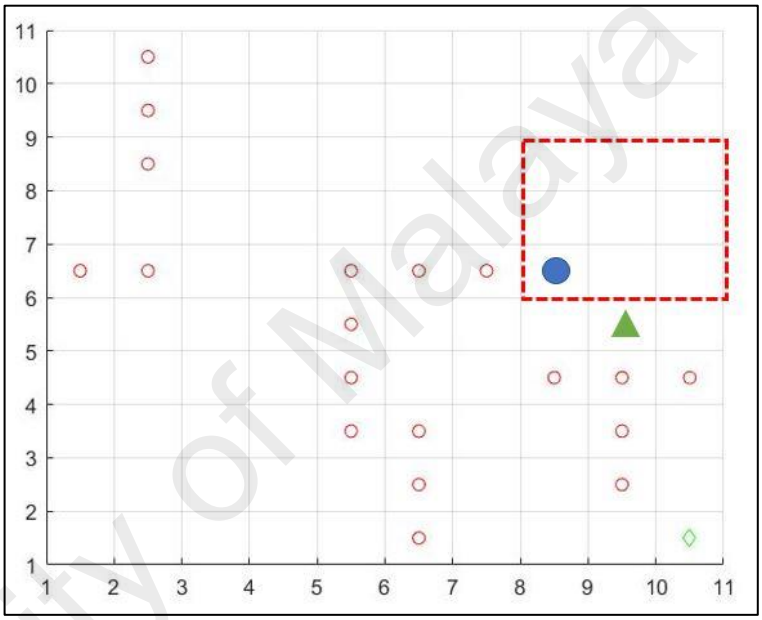

Figure 47: Temporary target located at bottom-right from current position

At this condition, the Windowing Grid should move downward by one row and the result after relocating the Windowing Grid can be found in Figure 48.

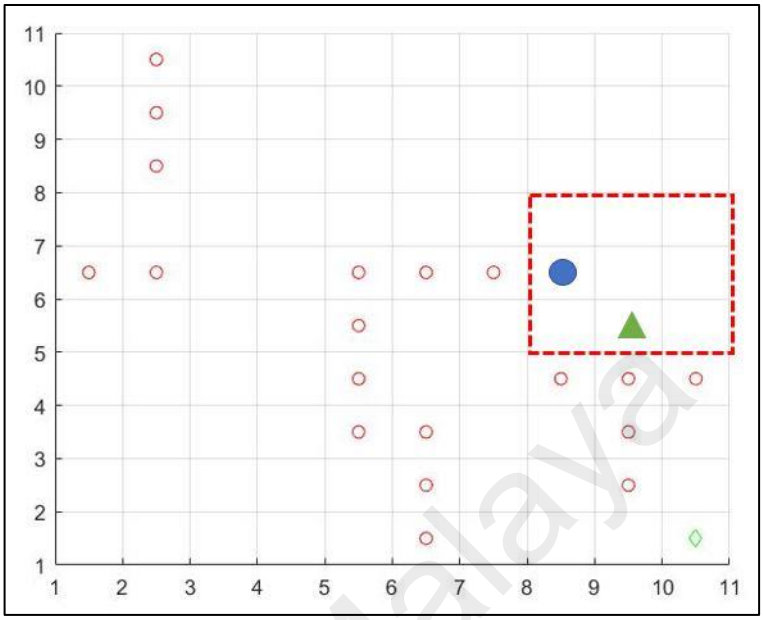

Figure 48: Position after Windowing Grid is shifted down by one row

vii. Current position lies at w32

When the current position of robot lies on cell w32 of Windowing Grid, there are three states of the temporary target that possible to happen.

W11	W12	W13
W21	W22	W23
W31	W32	W33

Table 10: The Windowing Grid variable with shaded cell at w32

The three states of the temporary target could be affected the Windowing Grid position as mentioned in the figure below. For first condition when the current position lies on w32 cell, it's when the temporary target is located at bottom-left of it. Like in Figure 49 (a), temporary target is located at (7,5) and the current position is at (8,6) which is upper by one row and one column of the temporary target.

While in Figure 49 (b) the temporary target is just located at the bottom side of the current position, and the third condition is where the temporary target is located at the bottom-right of the current position shown like in Figure 49 (c).

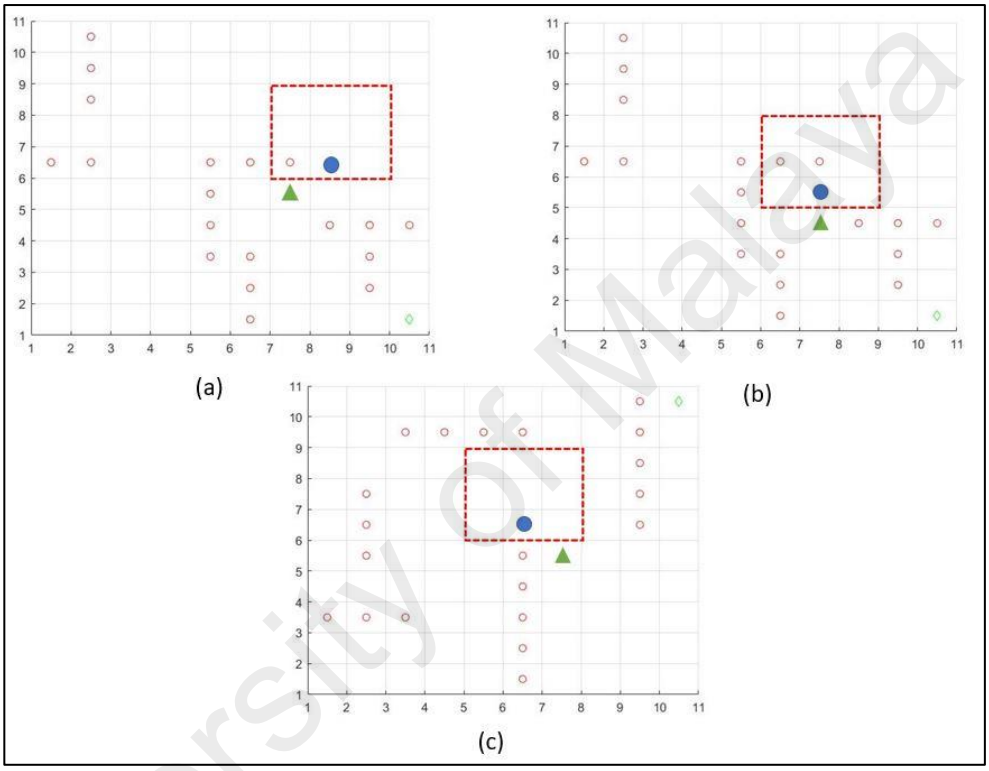

Figure 49: Three condition effected target that affected the transformation of Windowing Grid when the current position lies on cell w32

When all these three conditions are met, the Windowing Grid should shift down by one row.

All the conditions after shifting are shown in Figure 50 (a), (b) and (c).

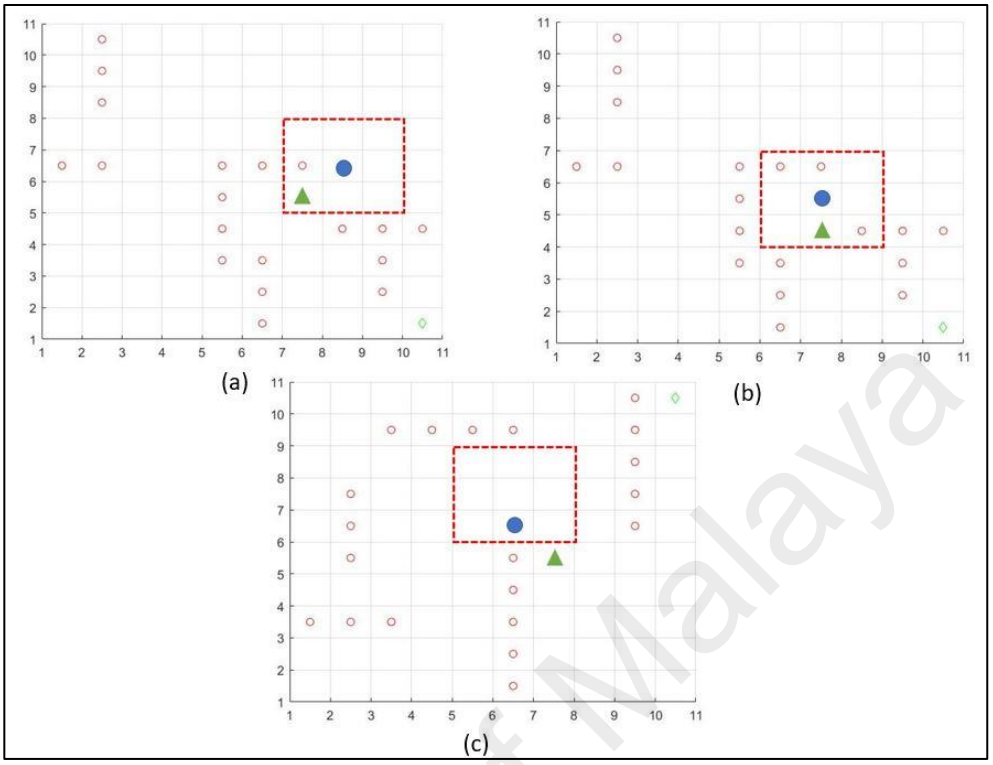

Figure 50: The resultant of transformation of Windowing Grid based on three conditions

viii. Current position lies on w33

At this position the robot is located on w33 cell of Windowing Grid like in Table 11. The shaded cell is where the current position of the robot is located.

W11	W12	W13
W21	W22	W23
W31	W32	W33

Table 11: Windowing Grid with shaded cell at w33

In this case, there are three conditions require the Windowing Grid to transform. There are like following cases:

Case 1: Temporary target is located at bottom or bottom-left of current position

In Figure 45 (a), the location of the current position is at (8,6) while the temporary target is located at (7,5), which lower by one column and one row.

Different with the location of the temporary target shown in Figure 51 (b) where it is just below one row of the current position.

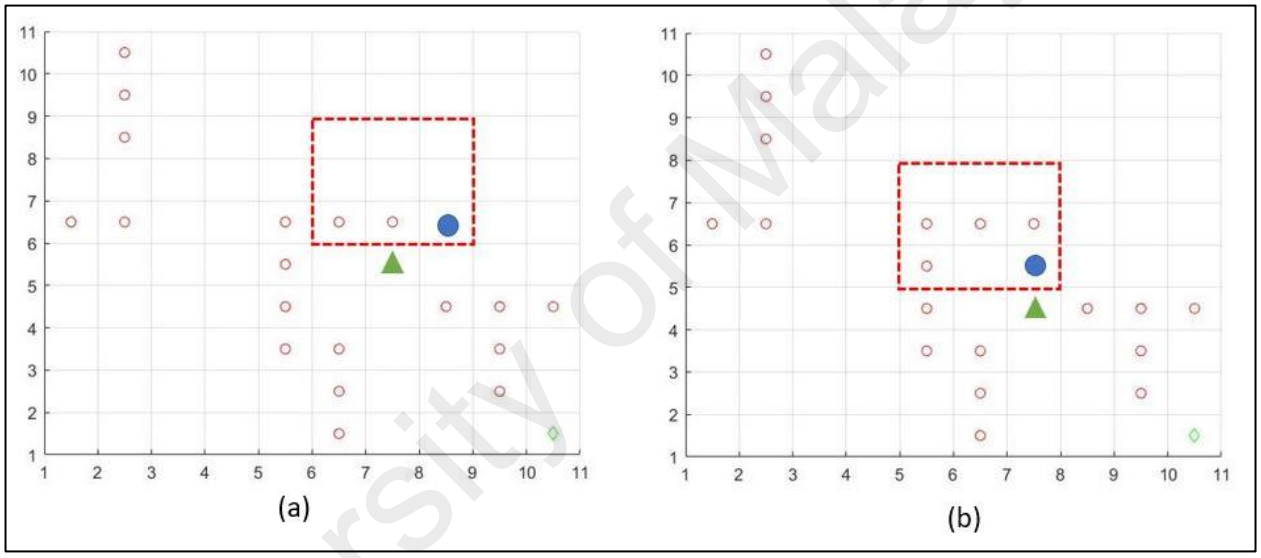

Figure 51: (a) Temporary target is located at bottom-left from current position, (b) Temporary target is located at bottom of current position

These two conditions require the Windowing Grid shifted to downward by one row. The resulting of transformation of these two conditions can be found in Figure 52 (a) and (b).

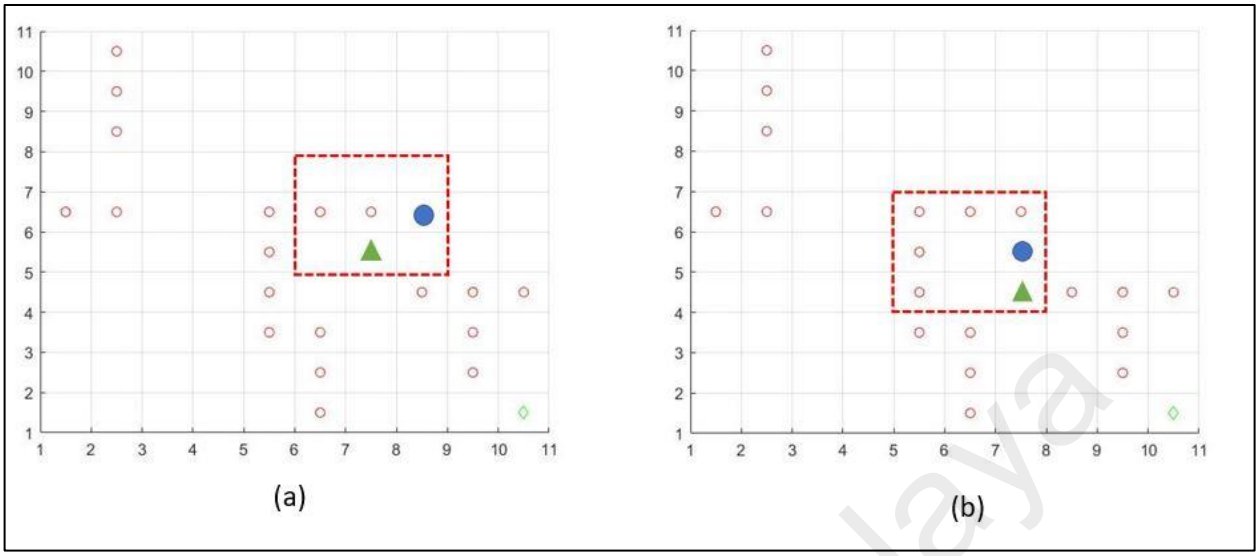

Figure 52: The last position of Windowing Grid after transformation

Case 2: Temporary target is located at the bottom-right of current position

This is the only situation that happens when the current position is located at cell w33 of Windowing Grid. Figure 53 (a) shows that the temporary target is located at bottom-right from the current position. It is one column upper and one row lower than current position.

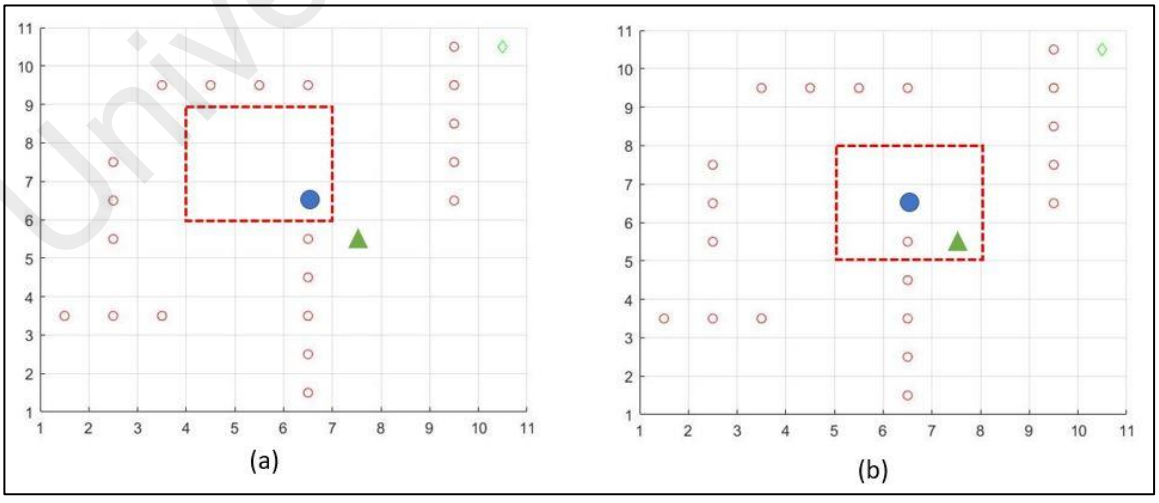

Figure 53: (a) Temporary target is located at bottom-right from current position, (b) The position of Windowing Grid after transformation

The resultant of this case can be found in Figure 54 (b) where the Windowing Grid is shifted in diagonal way.

Case 3: Temporary target is located at the right side or top-right of current position

For these two states where the temporary target is located at the right side or top-right of the current position. This condition requires the Windowing Grid to move to the right by one column.

The example of first state is where the temporary target is located at left side can be found in Figure 54 (a) while Figure 54 (b) shows the position of temporary target is located on the top-right of the current position. Figure 55 shows the Windowing Grid is transformed by shifted to the right by one column.

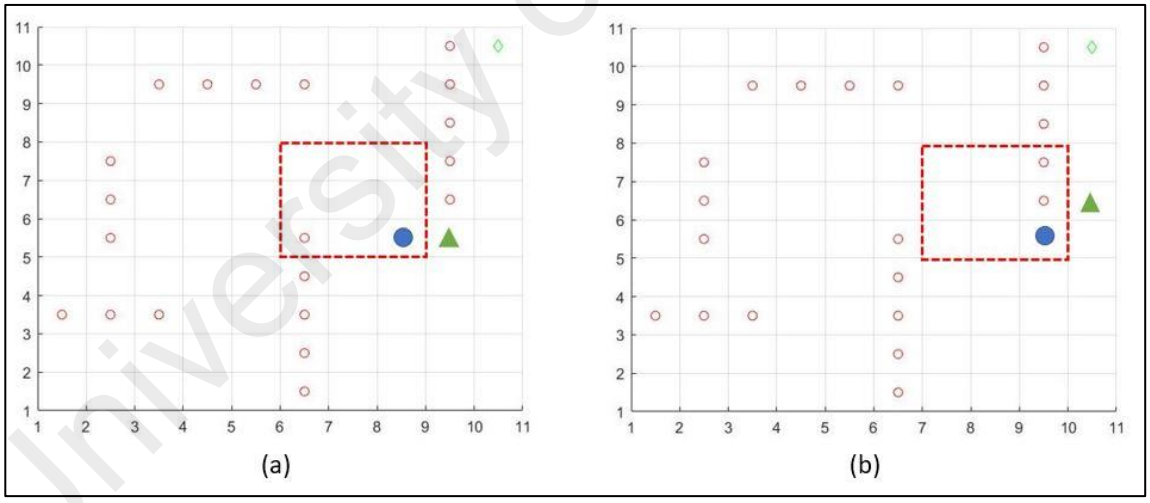

Figure 54: (a) Temporary target is located at right side from current position, (b) Temporary target is located at top-right from current position

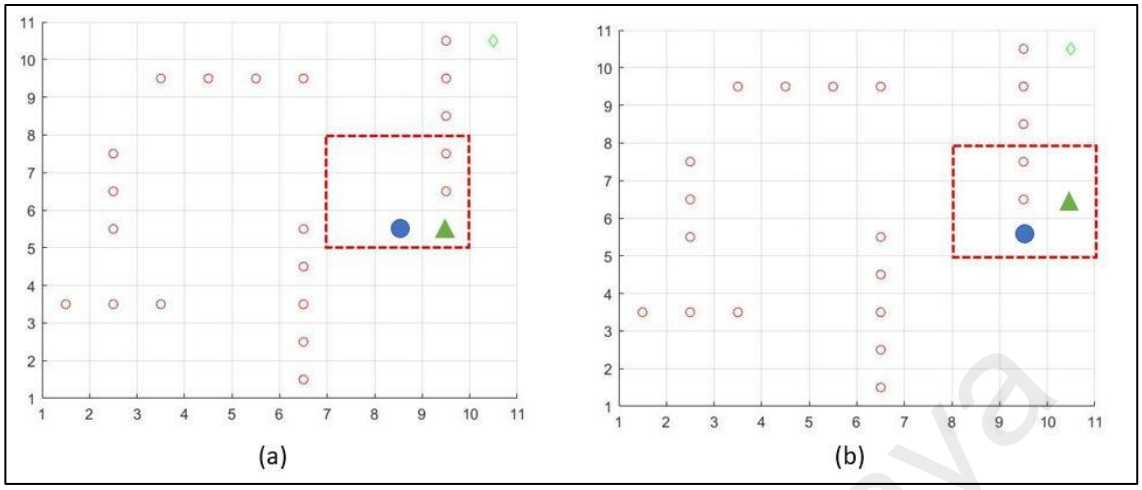

Figure 55: The location of Windowing Grid each of the states after transformation

CHAPTER 4

RESULT AND DISCUSSION

4.1 Network Training Accuracy

Figure 56 shows the training progress of the CNN. The system is in process of learning and adjusting the backpropagation value from epoch 0 until 50. After epoch 50, the system looks stable. Training time took 20 seconds.

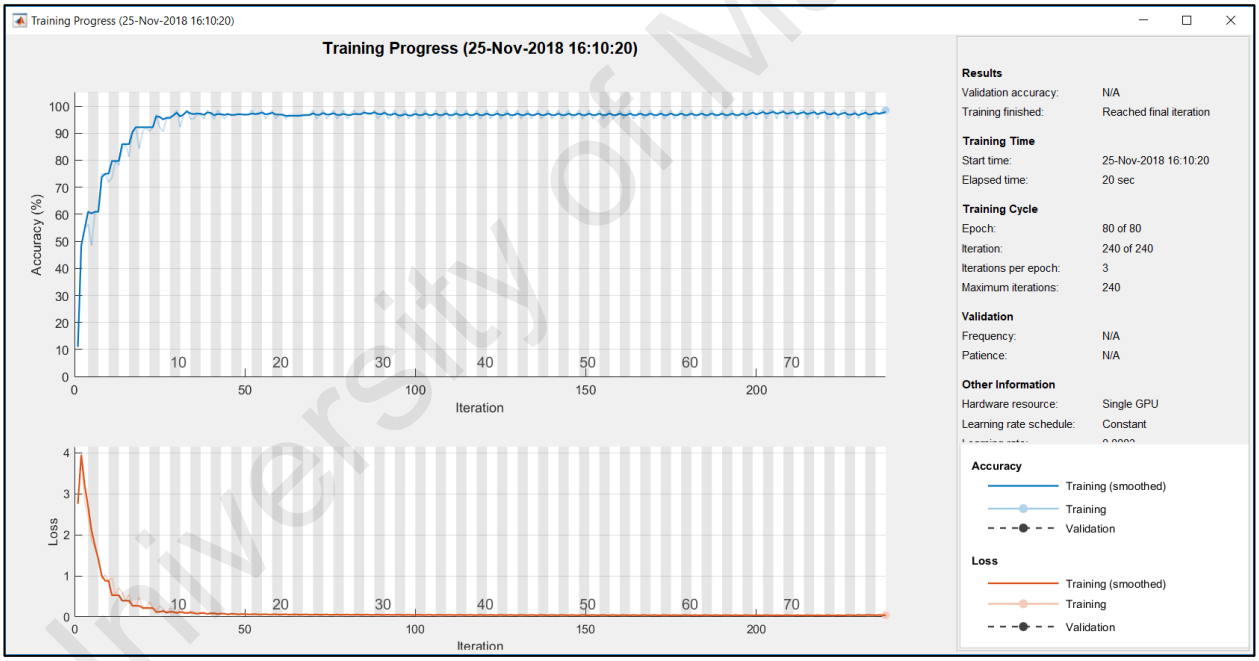

Figure 56: Plotted accuracy of the network

4.2 Path Planning by Robot in Map 1

Map 1 is the lowest level of difficulty and it is unique with its own characteristic differing from the other two maps. Figure 57 shows the environment of Map 1. The start position is located at (1,1) and the robot should reach target at (10,10). By

referring to the temporary target which assigned by A* Algorithm, the robot is able to reach the target by taking 14 steps of movement. The movement list is shown in Diagram 5.

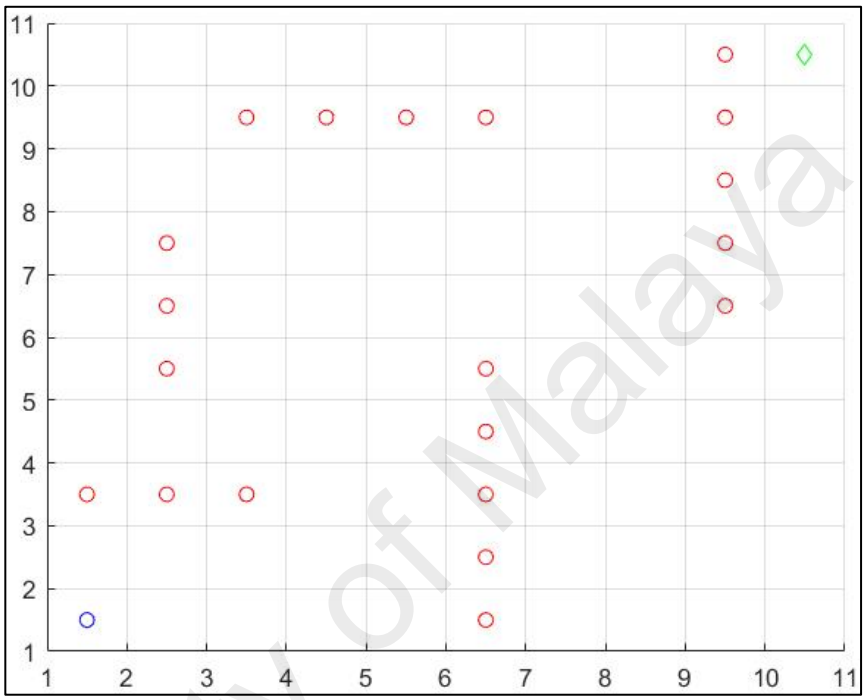

Figure 57: 2D environment of Map 1

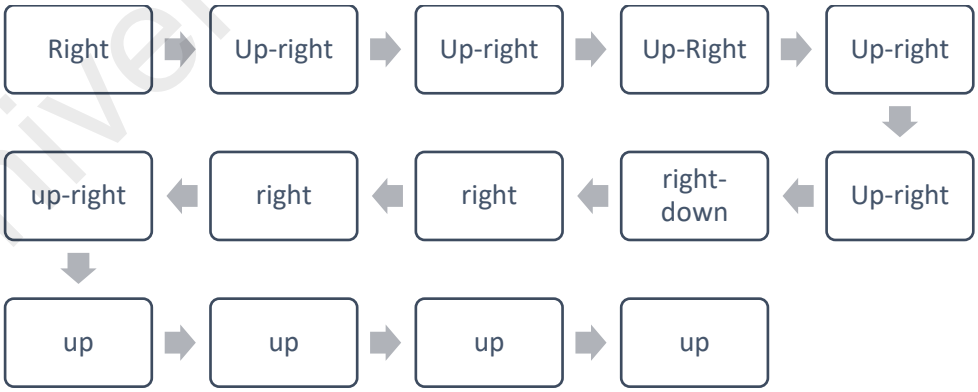

Diagram 5: The flow of movement taken by robot in Map 1

Figure 58 shows the result of Windowing Grid in terms of current position and temporary target assigned by A* algorithm.

Figure 59 and Figure 60 show every step taken by the robot to reach the target position which is located at the cell (10,1).

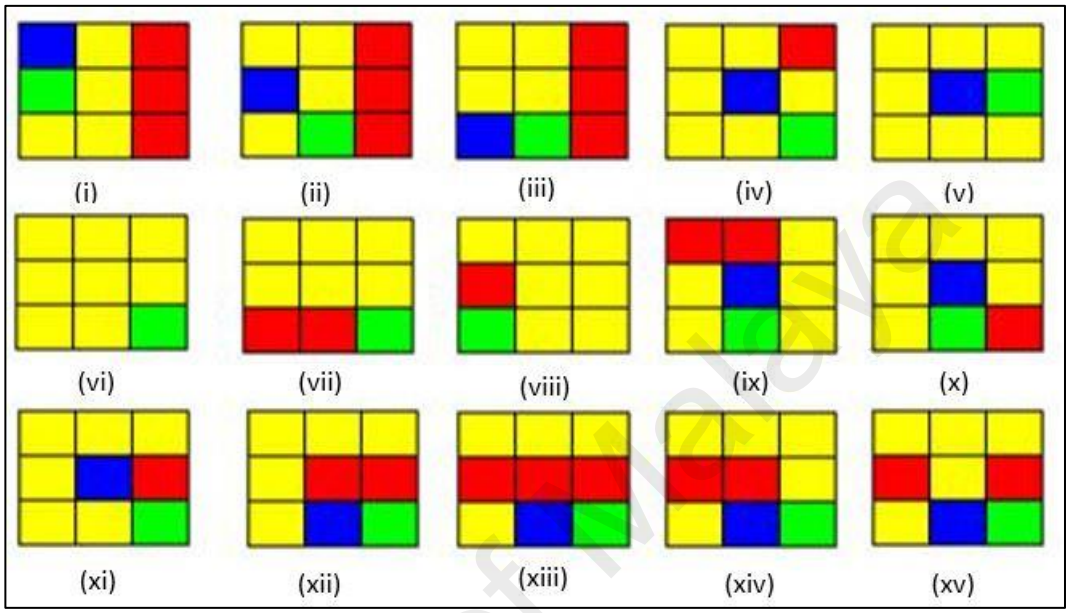

Figure 58: Result of Windowing Grid of Map 1

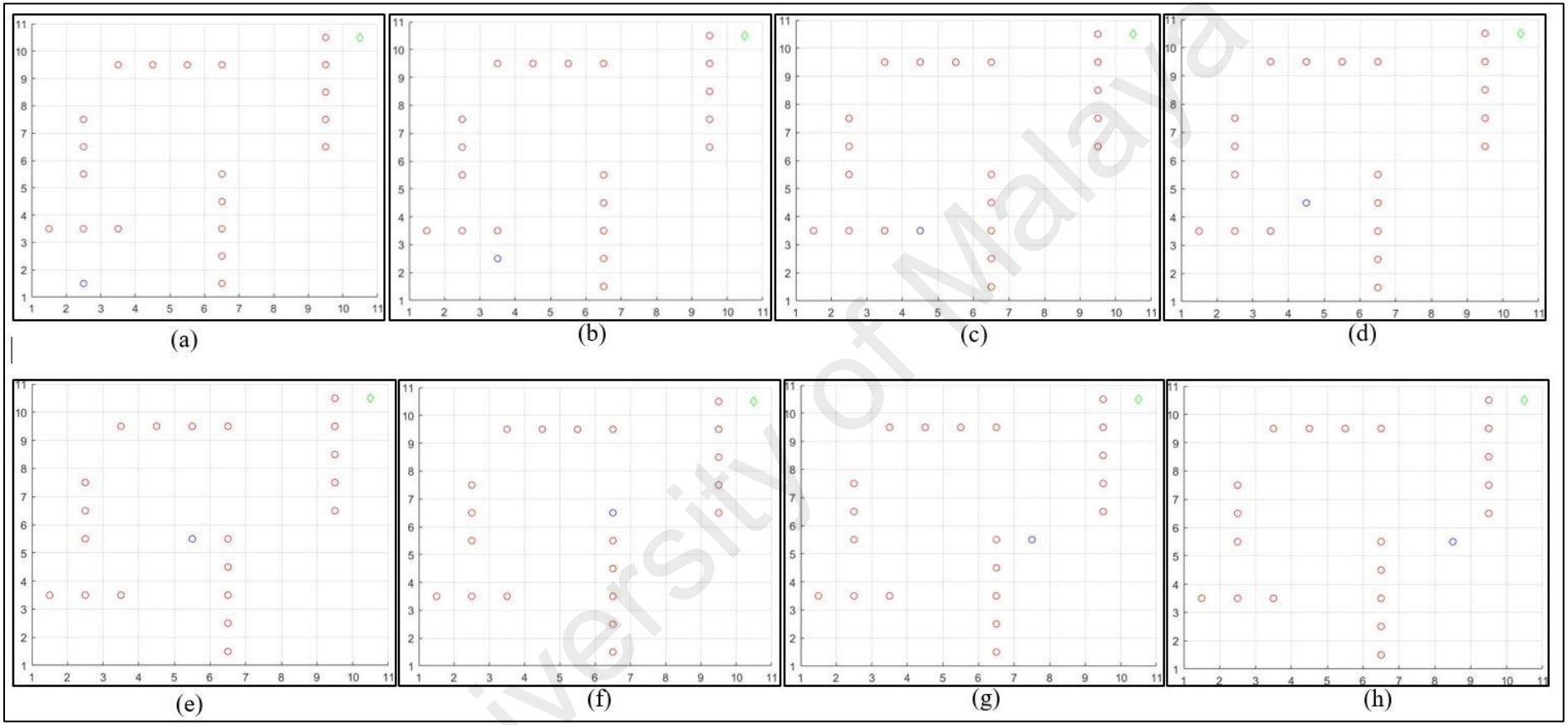

Figure 59: Current position of the robot in Map 1

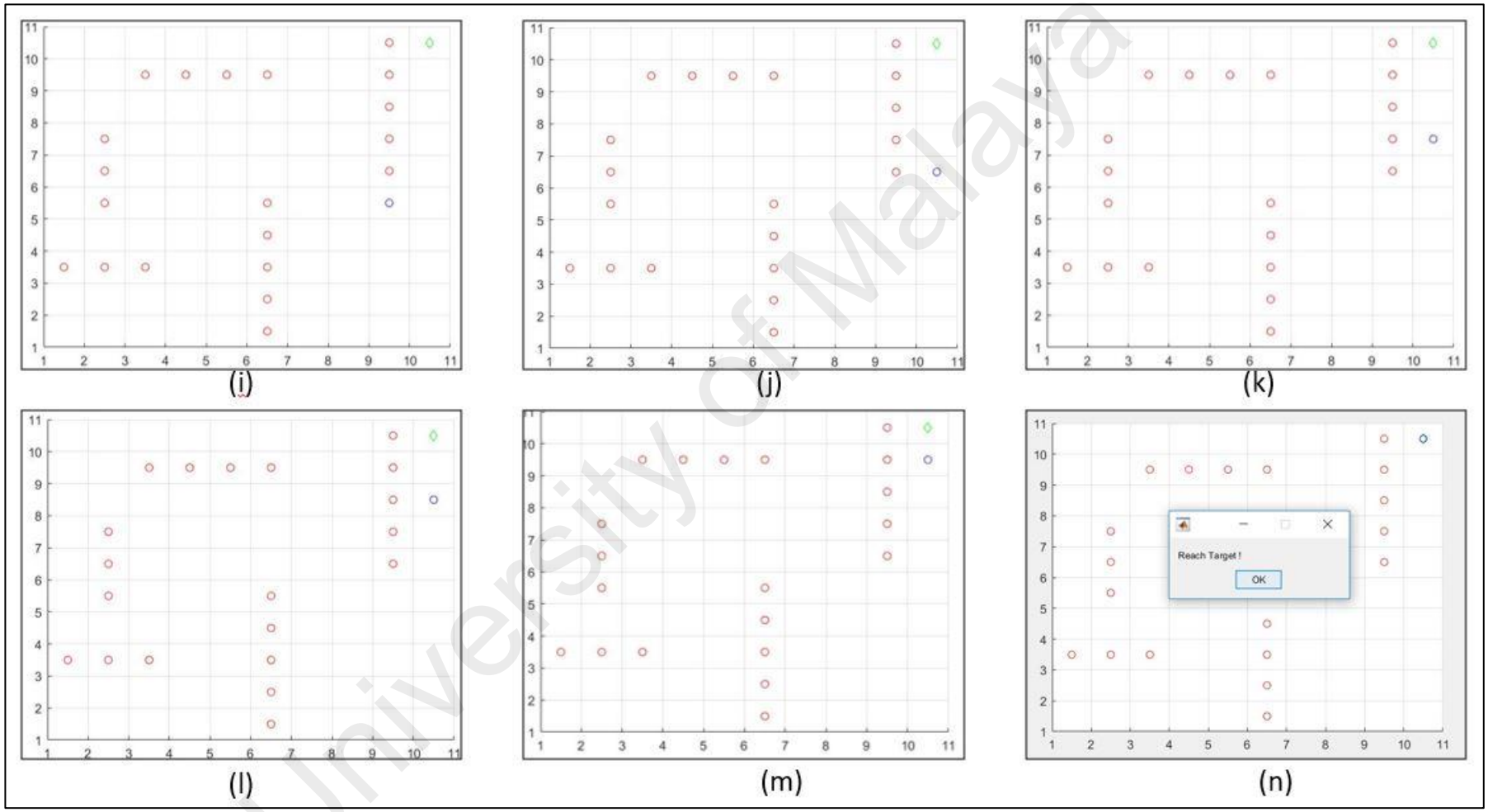

Figure 60: Continue

From Diagram 5 and Figure 58, the network is able to reach the target position which is located at point (10,10) by taking 14 steps. In Figure 58, step number 3 and number 5 predict with less accurate from it supposed direction but the network is able to gives prediction on safety direction and able to redirect to the corrected path. The robot able to pass the first obstacle located in horizontally at y=3. Obstacle 2 where they were placed vertically at x=6. The robot was able to climb and can pass over the obstacle. The third obstacle where the robot should move along the edge of the frame. Less accuracy and bad information about current position would make the network to make prediction until the robot can move out from the map frame. At this obstacle, the robot was able to control and move up without failing and hitting the obstacle.

4.3 Path Planning by Robot in Map 2

Map 2 is of medium difficulty level where it has three groups of obstacles. The initial position of the robot is located at (1,10) and the target position the robot should reach is at (10,1). The obstacles are placed by grouped. Diagram 6 shows the movement taken by the robot start from the initial position to the target location.

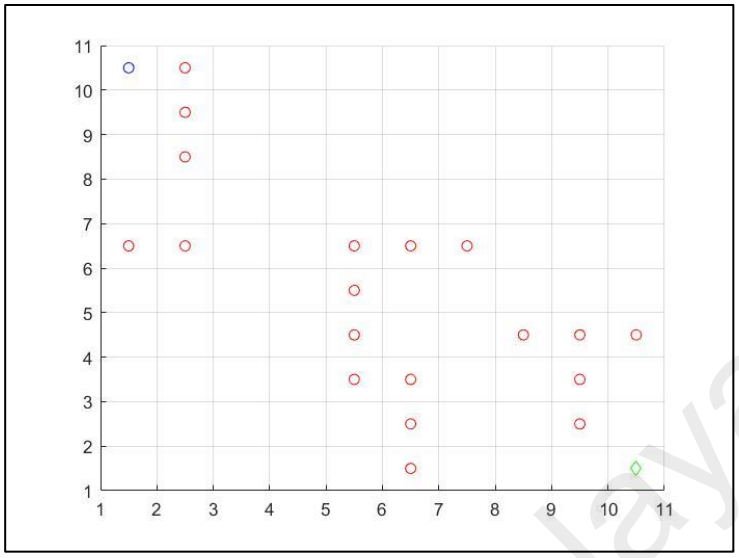

Figure 61: Map 2 environment

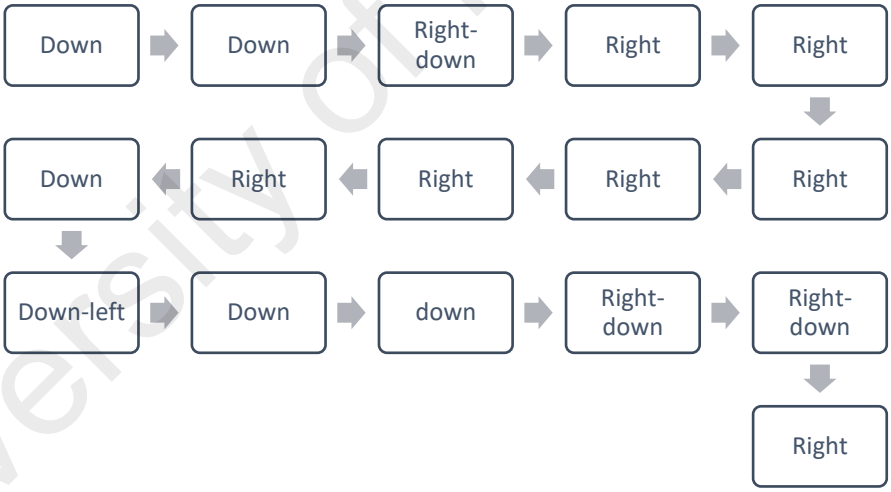

Diagram 6: The flow of movement taken by robot in Map 2

Figure 62 shows that the current position of robot at each movement taken.

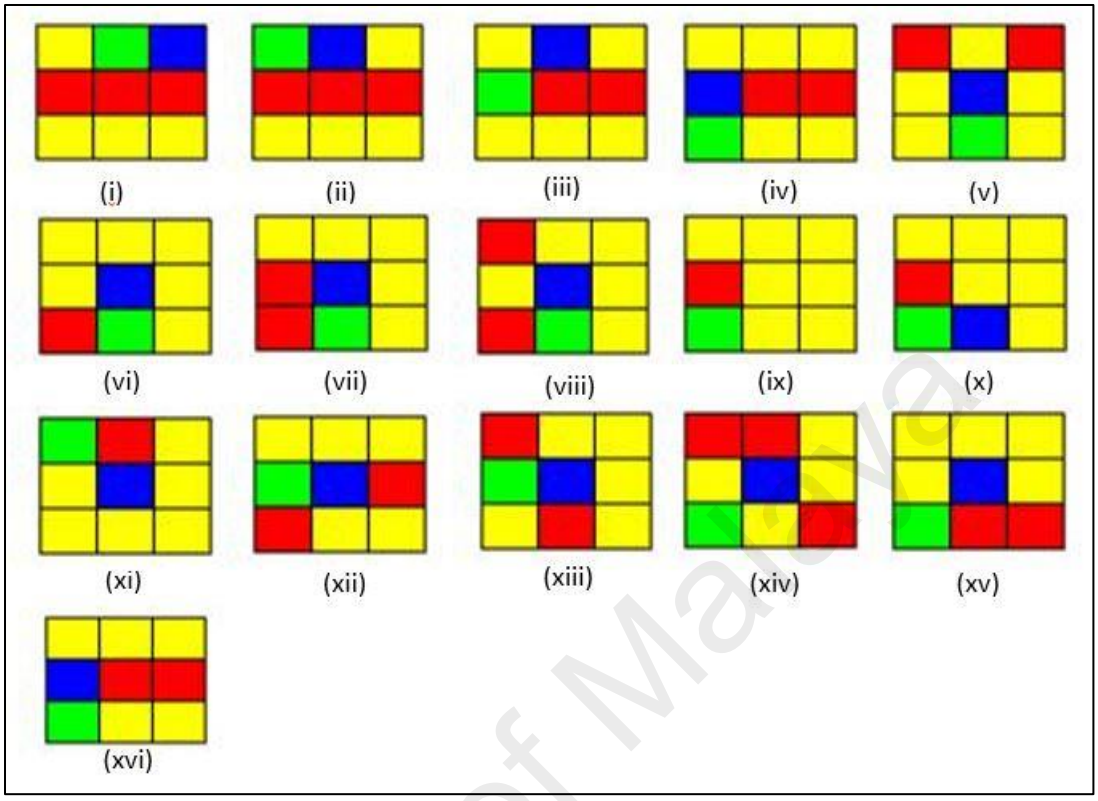

Figure 62: Result of Windowing Grid of Map 2

For this environment map, the robot creates 16 steps to reach the destination location. The shortest steps are 15 steps. The robot is less accurate in predicting at step 9 which is it supposed to take the right-down direction but the robot moves to the right. Thus, in order to turn it down, one extra step is taken to the down direction. During the process, the network works by producing zero prediction hitting the obstacle and no out of frame prediction.

The robot is able to move down at the initial stage where it moves along x=1. Here the frame limit is located on the left side of the robot and the obstacles are on the right side of it. The network is able to give correct predictions to pass this obstacle. The sudden end of the obstacle 1, there are obstacle in front of the robot where the robot should turn to right to avoid collision.

In Figure 63 (c) and (d) is where the area that the robot should pass and avoid the obstacle. There is an open area between obstacle 2 and 3. This is purposely to test if the robot will turn the direction to down. In Figure 63 (e) and (f) shows that the robot is able to follow the correct path.

Next situation is like in Figure 63 (i), (j) and (k) where it moves along the horizontal obstacle, and able to turn down and makes cornering to pass the L slot in Figure 63 (l), (m) and (n). This situation is opening to the critical part where the robot should turn again to the right so it can reach the destination position at x=10 and y=1.

From Figure 63 (j) to (q) the path looks like mirrored ‘S’ path where it has 2 times to turn. The robot is able to make the turning without hit the obstacle and able to do the high accuracy prediction.

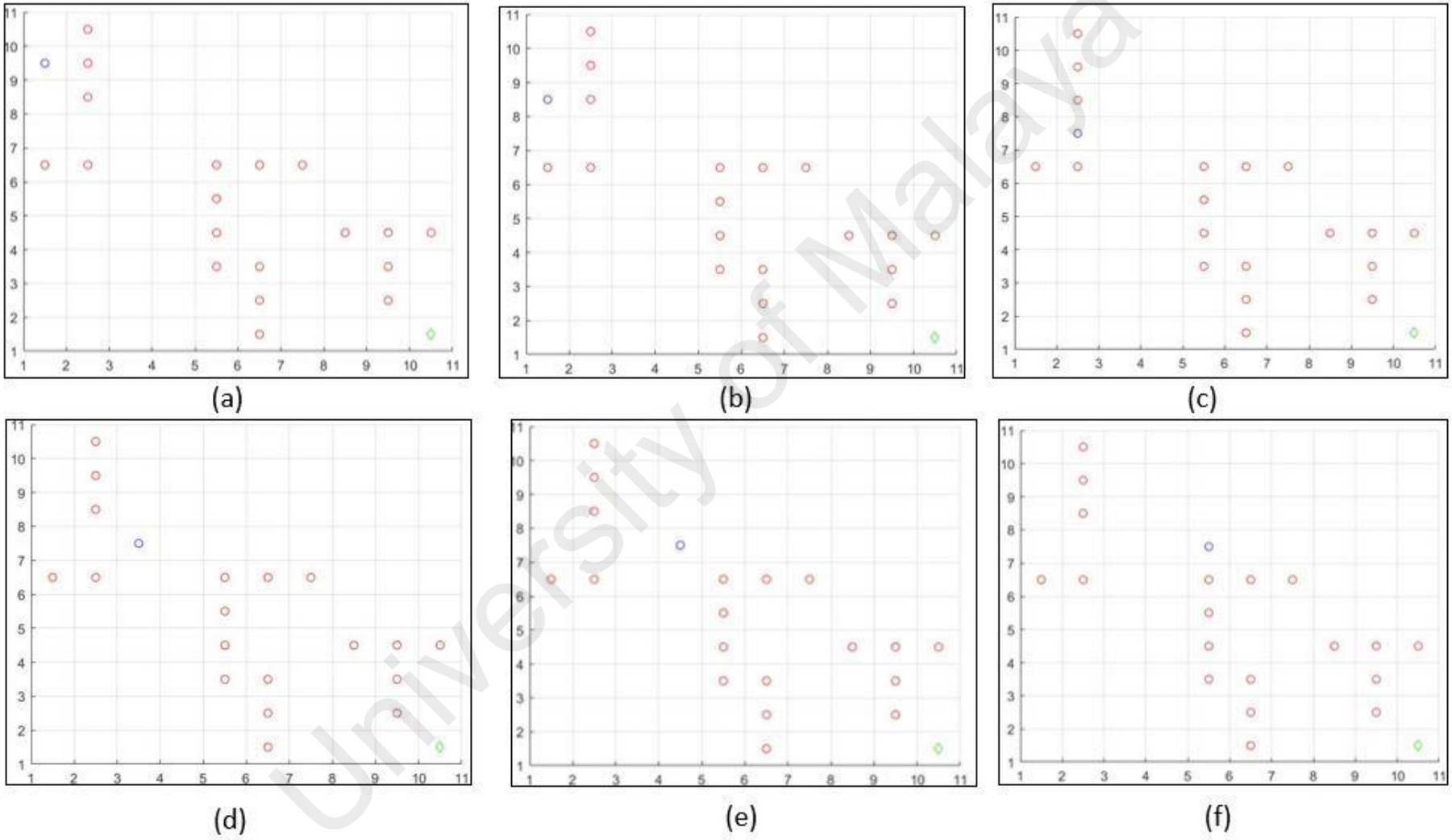

Figure 63: Current position of the robot in Map 2

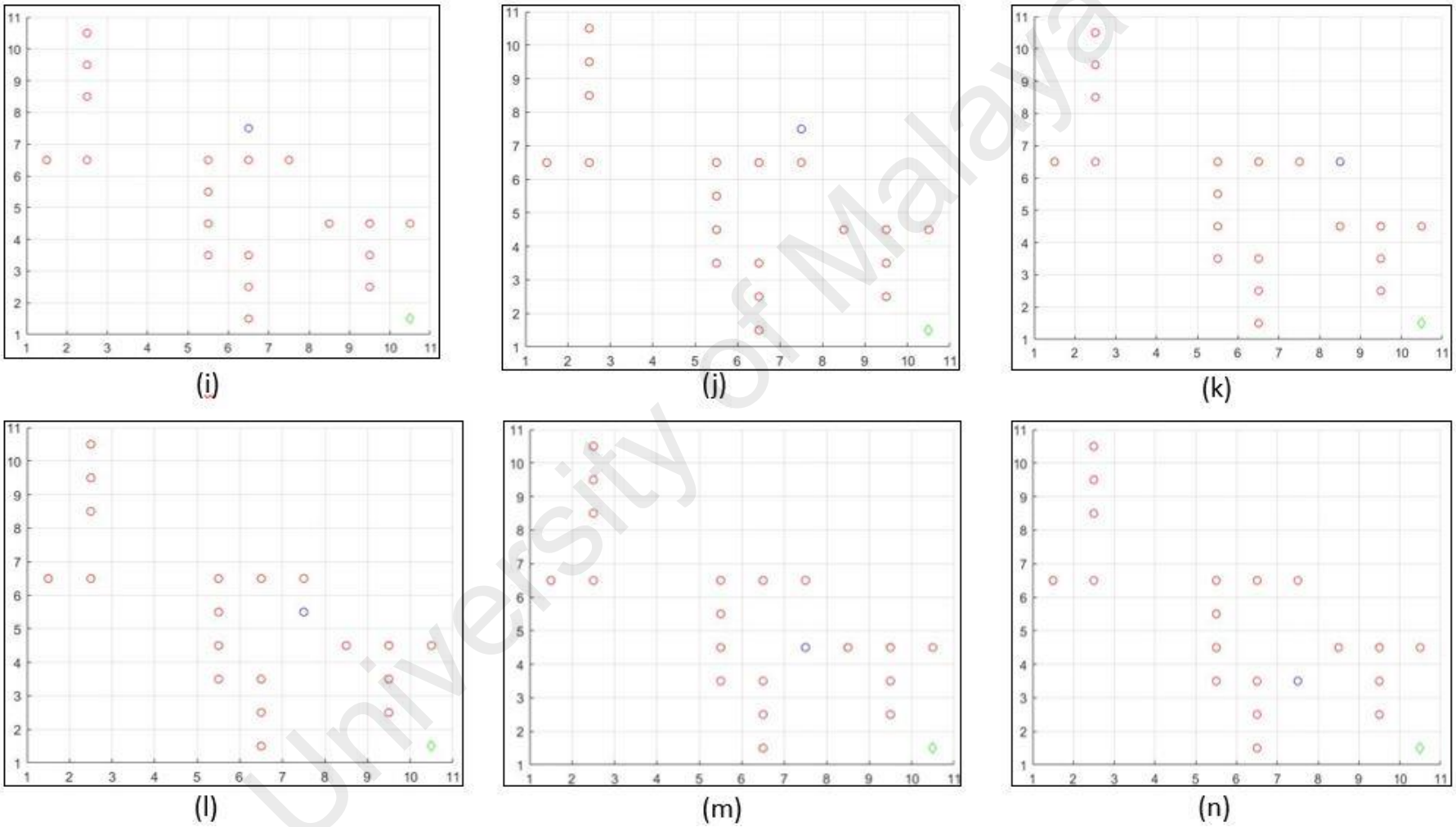

Figure 64: Continue

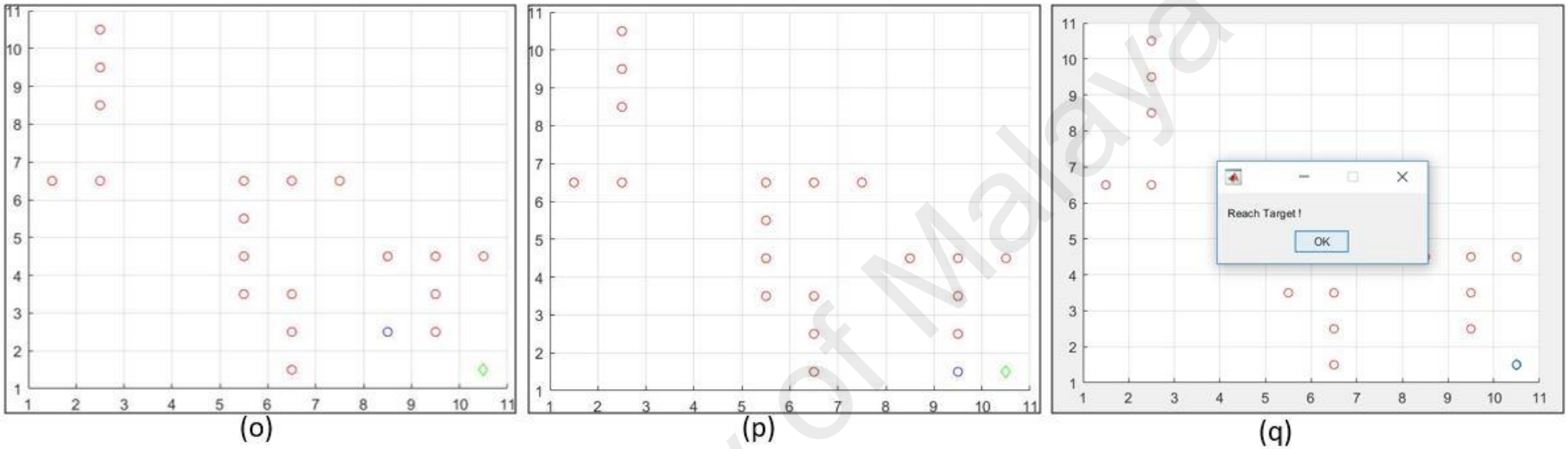

Figure 65: Continue

4.4 Path Planning by Robot in Map 3

Figure 66 shows the environment map for Map 3. The robot starts at point (10,10) and the destination is located at point (3,4).

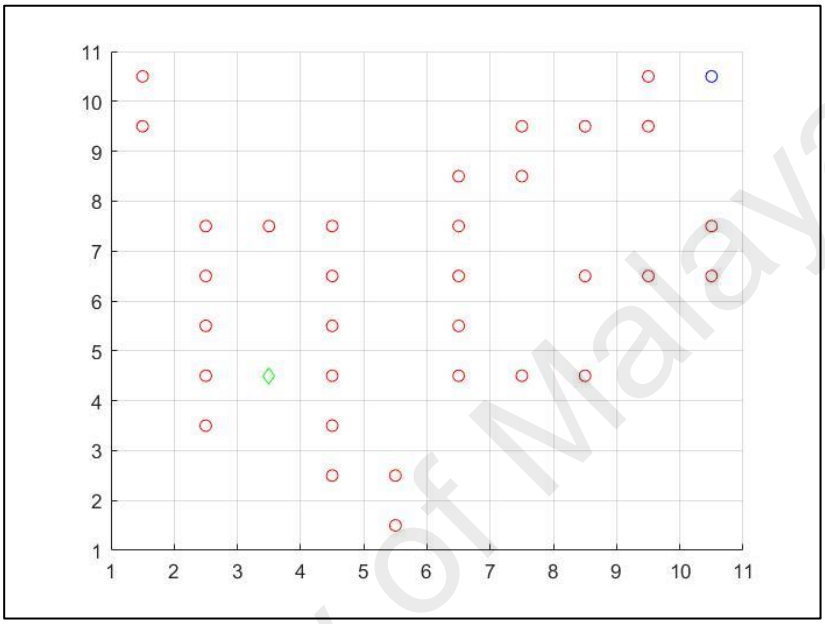

Figure 66: Environment map for Map 3

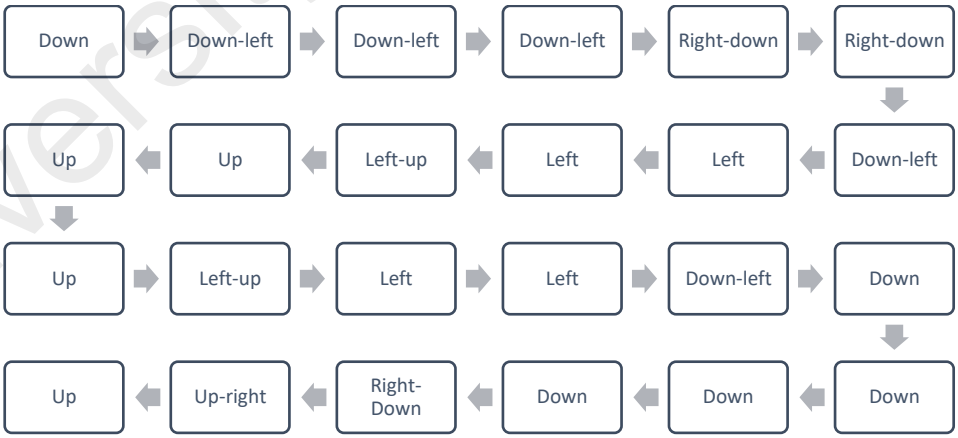

Diagram 7: The flow of movement taken by robot in Map 3

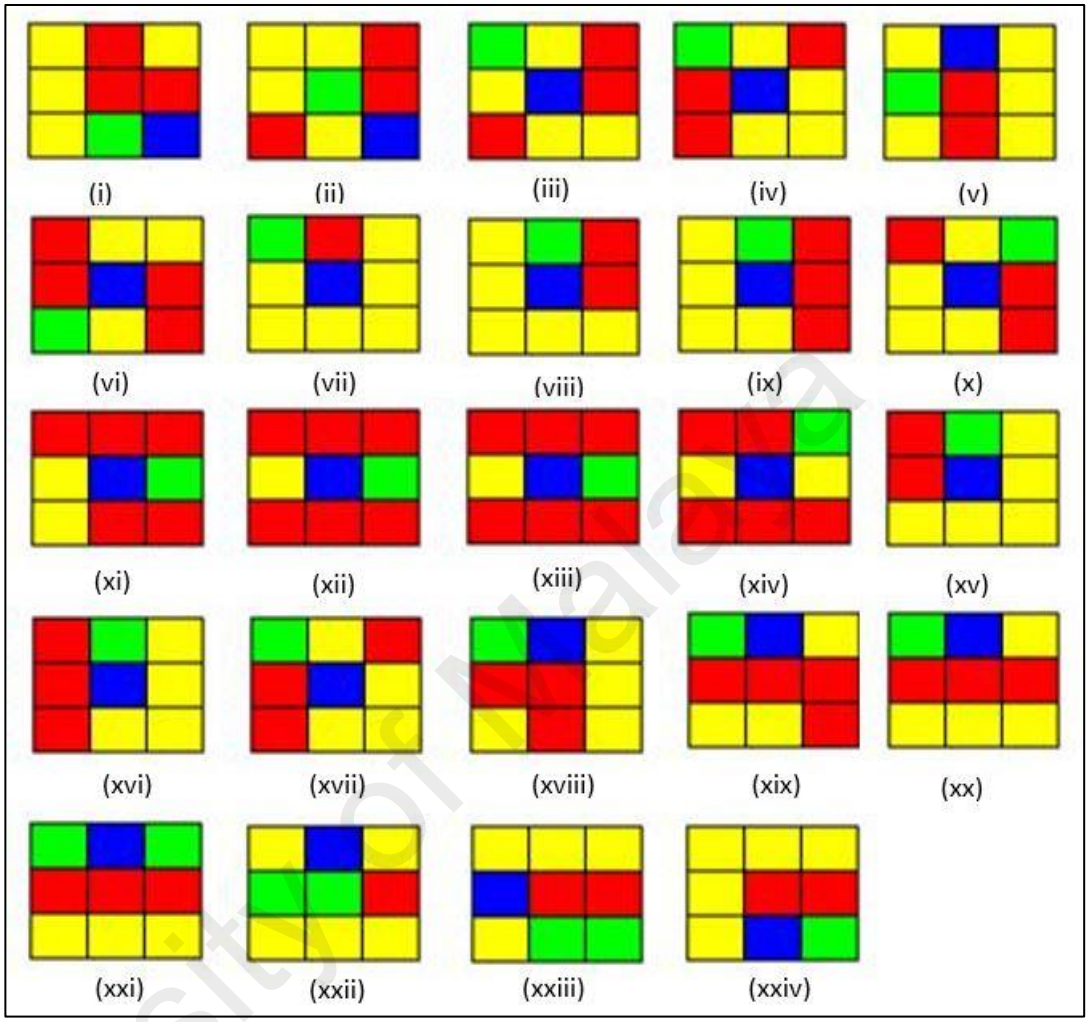

Figure 67: Result of Windowing Grid of Map 3

The robot takes 23 steps of direction to reach the destination position at (3,4). At first obstacle, the robot is able to move down along the map's frame on the right side and the obstacle is on left side. It has succeeded to move diagonal direction like in Figure 68 (b) until (d) and do the slight corner like in Figure 68 (e) until (g).

Then the robot moves along the obstacle and enter the narrow route. Now the obstacle is at the left and right of the robot. It should take up direction for 4 steps and move to the left when it reaches the end of the obstacles.

After moving to the left of the map, the robot again moves down for 5 steps and turns to the right at the end of the obstacle which is at location (2,2). The robot should avoid the obstacle from hitting them and avoid from making prediction which makes the wrong direction that allow the robot from moving out from the map frame.

During the progress, the robot has succeeded to pass through this area without hitting the obstacle and not going out of the frame.

The last part is that the robot needs to move up by 2 steps to reach the target position. Here the robot makes one wrong prediction, but is still safe and can re-route to the original path.

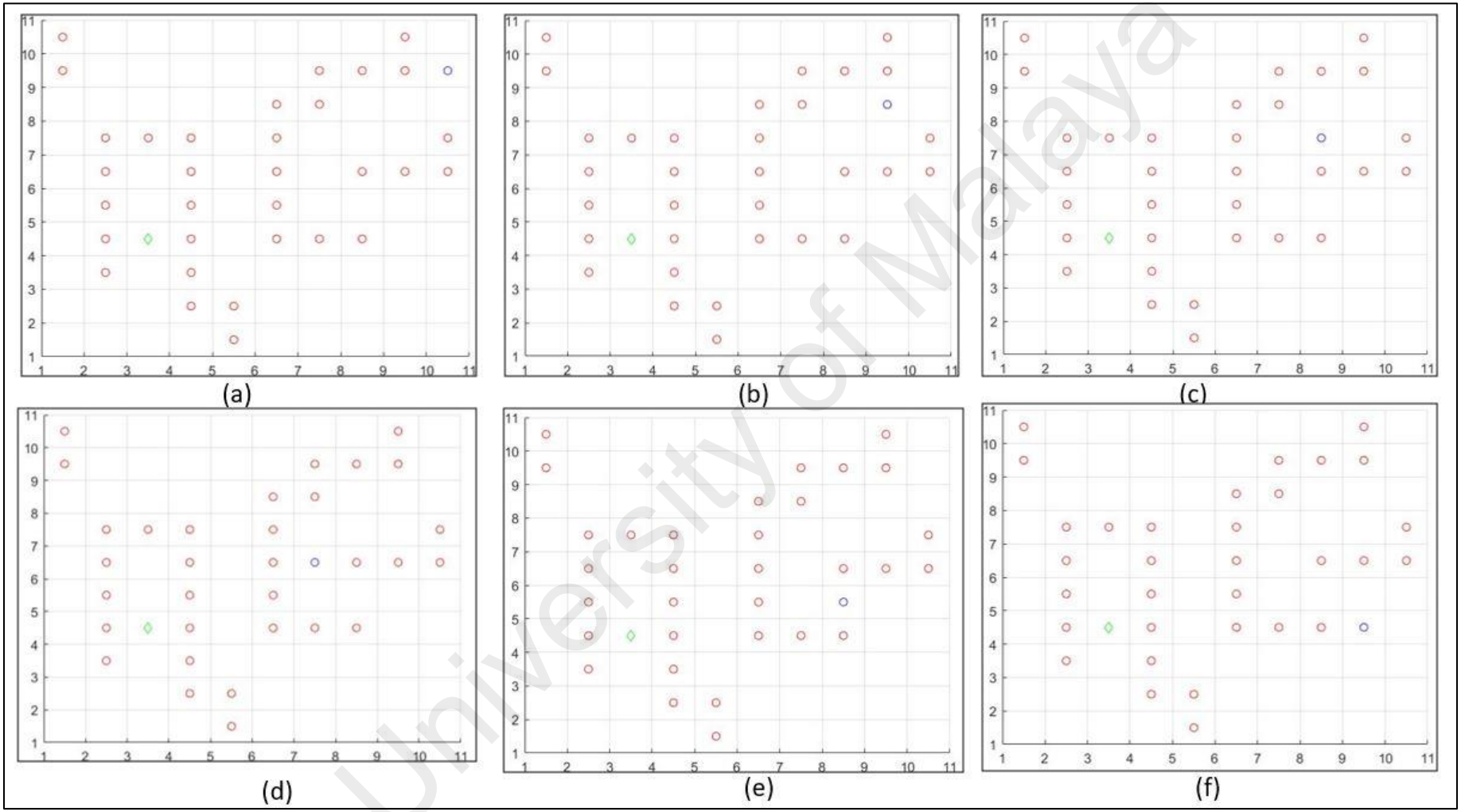

Figure 68: Current position of the robot in Map 3

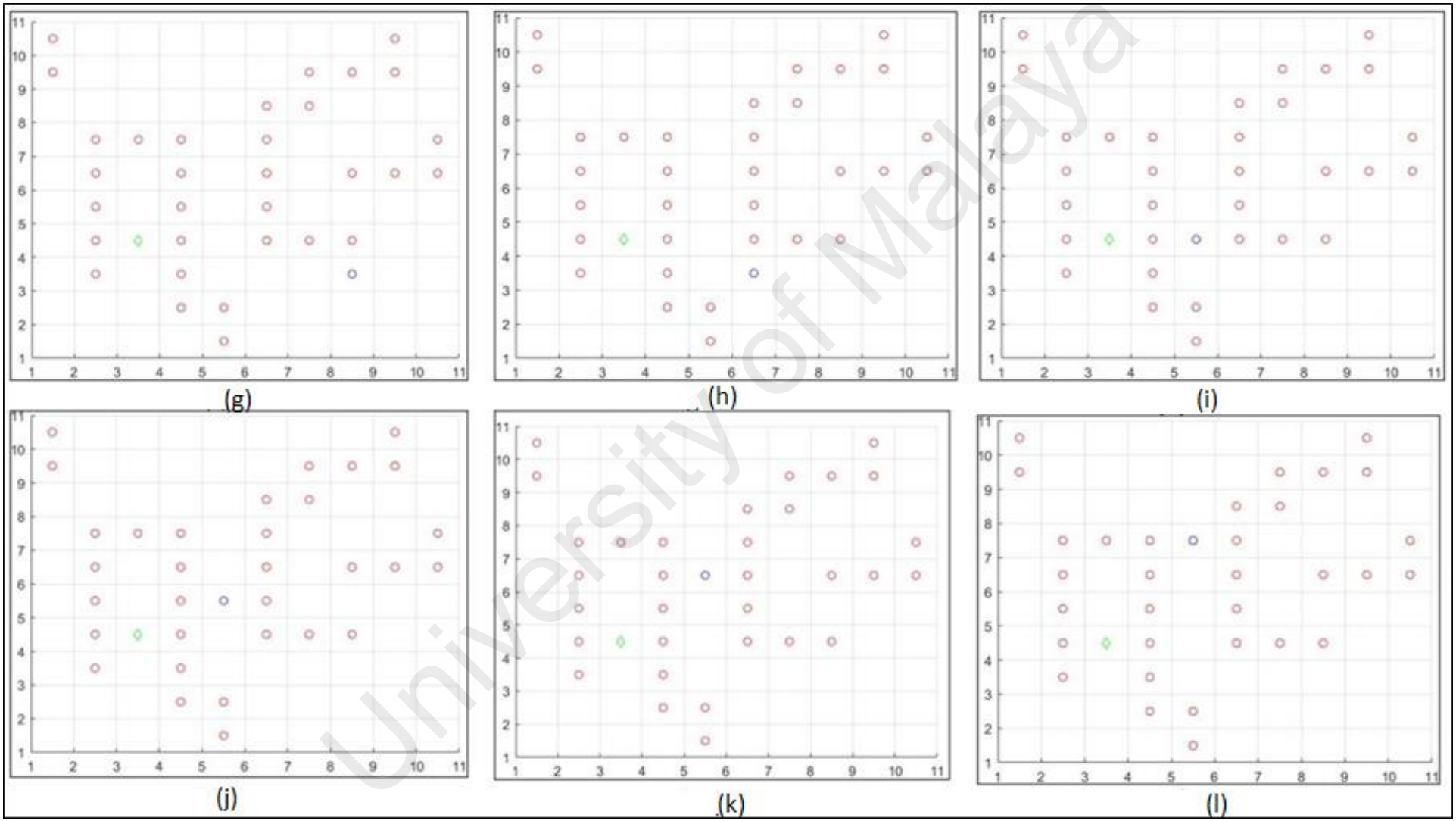

Figure 69: Continue

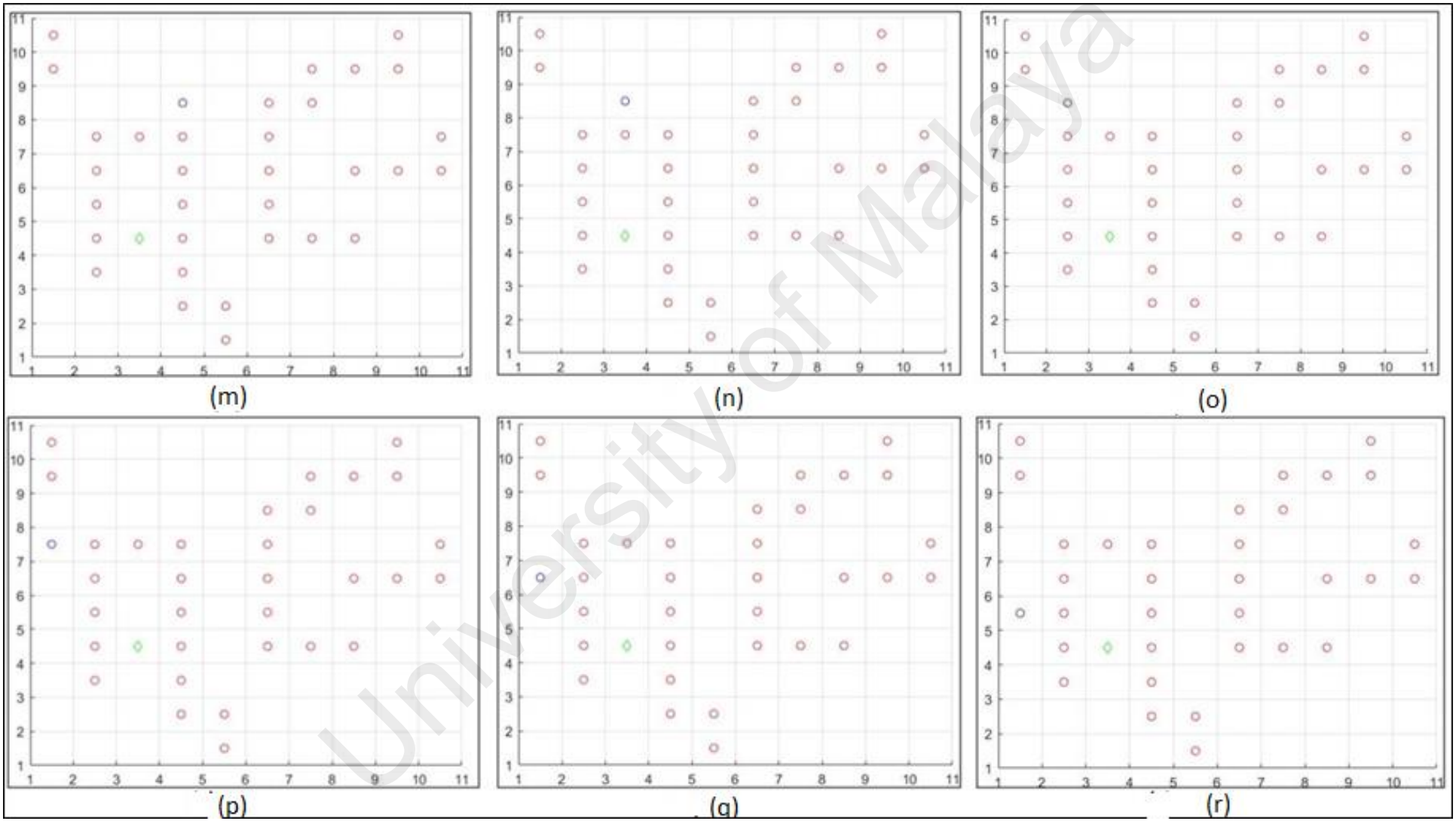

Figure 70: Continue

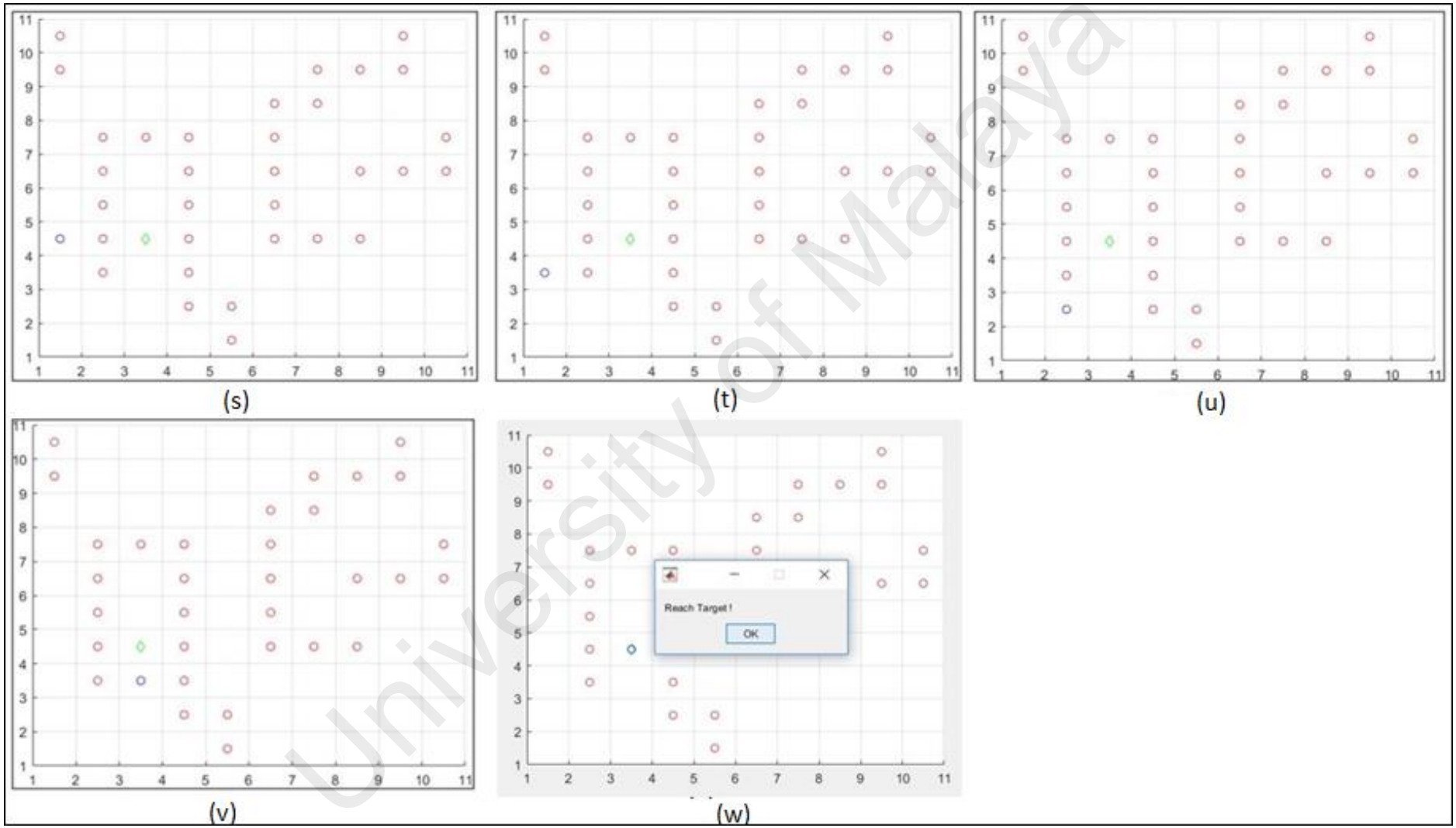

Figure 71: Continue

CHAPTER 5

CONCLUSION AND FUTURE WORK

Throughout the project, the CNN is able determine and makes a high accuracy prediction for the given current position of the robot in the MATLAB simulation window. The model was able to understand the eight different directions from the minimum data train. Originally, the project should be fed with hot encode value which represent each of the directions. The list of 4D arrays of the matrix prepared as training data to the network. Unfortunately, the CNN cannot produce a high accuracy. The accuracy is lower than 20%. Since this happened, the project changed the direction by providing the image as training and also the input for testing. From here, the accuracy of the networks rose around 95%. A total of 3 maps of environment are provided created in the MATLAB software and it is able to guide the virtual robot moving to the targeted position. Some arrangement of the obstacles is placed on purpose to evaluate if the model can avoid them successfully. Another challenge is the model require to give correct predictions when the robot is at the edge of the frame. Retraining of the network was included in the code when the robot hits the obstacle, and the robot needs to start from initial position. The percentage of re-training the network is very low and it happened around 5%. The correct input data to the network is the most important part. Preparing the data for input is really critical. All the possible directions taken and where the current position is located are very important to the system to collect the correct data input.

As the working environment created in the MATLAB is in grid coordinate, only static obstacles only can be evaluated. In future works, the dynamic obstacles can be included in the environment. The environment itself can be improved like a capturing video from machine vision and the robot can predict the safe and correct way it should take. In the other hand, the advanced CNN for path planning is being introduced which called Deep Q-Learning where it has higher accuracy in planning the path of the robot.

REFERENCES

1. Andrea, V., & Karel, L. (2015). MatConvNet. *MM '15 Proceedings of the 23rd ACM international conference on Multimedia* (pp. 689-692). Brisbane: ACM New York.

2. Anton, A. Z., & Ignat, R. E. (2018). *The Use of Convolution Artificial Neural Networks*.

3. Changlong, L., Chunyang, W., Bin, Z., Yongting, Z., Shun, F., & Haochen, L. (2017). *CNN-Based Vision Model for Obstacle Avoidance of Mobile Robot*, 1-4.

4. Christos, S., & Dimistrios, S. (n.d.). *Imperial College London*. Retrieved from NEURAL NETWORKS: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

5. Daphne, C. (2018, April 24). *An intuitive guide to Convolutional Neural Networks*. Retrieved from freeCodeCamp: https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050

6. Dominik, H. (1997). Fast Mostion Planning by Parallel Processing - A Review. *Journel of Intelligent and Robotic Systems*, 45-69.

7. Eike, R., Jannik, Q., & Christoph, S. (2017). *Driving Like a Human: Imitation Learning for Path Planning using Convolutional Neural Network*.

8. Eric, J., James, D., & Jacek, M. (1990). Layered Neural Networks with Gaussian Hidden Units as Universal Approximations. *Neural Computation*, 210-215.

9. Gohd, C. (2018, July 11). Retrieved from https://www.livescience.com/63023-sophia-robot-citizen-talks-gender.html

10. Hachour, O. (2011). Neural Path planning For Mobile Robots. *INTERNATIONAL JOURNAL OF SYSTEMS APPLICATIONS, ENGINEERING & DEVELOPMENT*.

11. Haykin, S. (2005). *Neural Networks, A Comprehensive Foundation.* Pearson Education.

12. Ines, G., Virgil, T., & Alexandru, G. (2009). *Mobile Robot Navigation based on CNN Images Processing – An Experimental Setup*.

13. Joel, O. G., Lucas, T. G., Amanda, C. D., Breno, Z., Paulo, D.-J., & Silvia, S. C. (2016). Vision-based Obstacle Avoidance Using Deep Learning. *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium*, (pp. 7-12).

14. Joseph, D., & Bradley, H. (2016). Neural Networks for Obstacle Avoidance.

15. Keith, S., & Wallace, L. (n.d.). *Deep Obstacle Avoidance*.

16. Khan, N. (2017, April 11). Retrieved from https://www.quora.com: https://www.quora.com

17. Lei, T., & Ming, L. (2016). *Mobile robots exploration through cnn-based reinforcement learning*.

18. Lei, T., Ming, L., Member, & IEEE. (2015). LATEX CLASS FILES. *Deep-learning in Mobile Robotics - from Perception to Control Systems: A Survey on Why and Why not*.

19. Lei, T., Shaohoa, L., & Ming, L. (2016). A Deep-Network Solution Towards Model-less Obstacle Avoidance. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2759-2764). Daejeon, South Korea: IEEE.

20. Lei, T., Shaohua, L., & Ming, L. (2017). *Autonomous exploration of mobile robots through deep neural networks*, 1-9.

21. Lingyan, R., Yanning, Z., Qilin, Z., & Tao, Y. (2017). *Convolutional Neural Network-Based Robot Navigation Using Uncalibtared Spherical Images*, 1-18.

22. Mariusz, B., Davide, D. T., Daniel, D., Bernhard, F., Beat, F., Prasoon, G., . . . Karol, Z. (2016). *End to End Learninf for Self-Driving Cars*.

23. Markus, W., Dushyant, R., Dominic, Z. W., Peter, O., & Ingmar, P. (2017). *Large-scale cost function learning for path planning using deep inverse reinforcement learning*, 1-15.

24. Marr, B. (2016, Dec 29). Retrieved from https://www.forbes.com/sites/bernardmarr/2016/12/29/4-amazing-ways-facebook-uses-deep-learning-to-learn-everything-about-you/#2102bb20ccbf

25. Mattson, N. (2016). Classification Performance of Convolutional Neural Network.

26. No'e, P. H., Fernando, C., & Luis, M. (2018). *Leraning Human-aware Path Planning with Fully Convolutional Networks*.

27. Noha, R., & Wolfram, B. (2018). *Effective Interaction-aware Trajectory Prediction using*.

28. Patrice, Y., David, S., & John, C. (2003). *Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis*.

29. Pierre, S., Soumith, C., & Yann, L. (2012). Convolutional Neural Networks Applied to House Number Digit Classification. *21st International Conference on Pattern Recognition (ICPR 2012)* (pp. 3288-3291). Tsukuba, Japan: IEEE.

30. Punarjay, C., Klaas, K., Tom, R., Stijn, W., Tinne, T., & Luc, V. E. (2017). *CNN-based Single Image Obstacle Avoidance on a Quadrotor*.

31. Roy, G., Andrzej, K., & Stan, C. (1995). Neural Networks. *Neural Networks Dynamics for Path Planning and Obstacle Avoidance*, 125-133.

32. Samer, L. H., Rishi, K., & Chris, R. (2015). *Using Convolutional Neural Networks*.

33. Saurabh, G., James, D., Rahul, S., Jitendra, M., & Sergey, L. (2017). *Cognitive Mapping and Planning for Visual Navigation*.

34. Schnell, B. (2016). *A hybrid architecture for pathnding in dynamic environments.* Software Technology System.

35. Shichao, Y., Sandeep, K., Chen, M., Stephanie, R., Manuela, V., & Sebastian, S. (2017). *Obstacle Avoidance through Deep Networks based Intermediate*.

36. Stromgren, O. (2018). *Deep Learning for Autonomous Collision Avoidance*.

37. Valeri, K., & Jianli, Y. (2011). *Neural Networks Based Path Planning and Navigation of Mobile Robots*, 173-190.

38. Waseem, R., & Zenghui, W. (2017). Neural Computation. *Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review*.

39. Wilko, S., Javier, A.-M., & Daniela, R. (2018). The Annual Review of COntrol, Robotics, and Autonomous Systems. *Annu. Rev. Control Robot. Auton. Syst. 2018*, (pp. 187-210).

40. Yang, L., Shujuan, Y., Yurong, L., & Yuling, J. (2016). Assembly Automation. *A Novel Path Planning Method for biomimetic robot based on deep learning*, 186-191.