

AI-BASED LEVEL DETECTION AND OPTIMIZATION OF ASSISTIVE ROBOT MANEUVERABILITY

Siti Fauziah Toha
Ahmad Syahrin Idris
Abdur Razzaq Abd Halim

AI-BASED LEVEL DETECTION AND OPTIMIZATION OF ASSISTIVE ROBOT MANEUVERABILITY

Siti Fauziah Toha, Ahmad Syahrin Idris,
Abdur Razzaq Abd Halim



Logo
Name

The book AI-Based Level Detection and Optimization of Assistive Robot Maneuverability Is published by Centre for the Professional Development (CPD), IIUM.

Centre for Professional Development (CPD)
International Islamic University Malaysia
Jalan Gombak,
Selangor Darul Ehsan,
MALAYSIA
Tel: +603-6421 5914/ Fax: +6421 5915
Email: admin_cpd@iium.edu.my
Website: www.iium.edu.my/centre/cpd

First published in 2021

Publication © Centre for Professional Development, IIUM.

©

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise except brief extracts for the purpose of review without the prior permission in writing of the publisher and copyright owner from Centre for Professional Development, IIUM. It is advisable also to consult the publisher if in any doubt as to the legality of any copying which is to be undertaken.

National Library of Malaysia
Cataloguing-in-Publication Data

Online Teaching for the Emerging Online Teacher

AUTHORS: Siti Fauziah Toha, Ahmad Syahrin Idris, Abdur Razzaq Abd Halim

e ISBN 978-967-19026-5-3

1. AI-Based Level Detection 2. Optimization Of Assistive Robot Maneuverability

PREFACE

Assistive devices for blind and visually impaired people are one of the technologies that famous among the researchers. There are many devices has been developed by the researcher in order to aid the visually impaired people to move around. One of the challenges faces by the blind and visually impaired people is stair. Using traditional method, the present of the staircase cannot be detected in a safe distance. Rovision is one of the devices that has the capability to guide the user to move to desired place without hitting any obstacle or object. In mobile robot, sensors play important roles in guiding the robot by sending the data of the surrounding for the robot to execute the action. Camera and ultrasonic sensor are two sensors that use in the Rovision to navigate the robot in safe distance while detecting the staircase. Image processing is one of the best methods in detecting staircase. It has a capability to learn by its own using the training dataset. By training the dataset, the system be able to identify the staircase and the position inside the camera frame to help Rovision to maneuver and guide the user safely. Rovision also use ultrasonic sensors to avoid obstacle in surrounding in order to have clear and safe path for the visually impaired person. This AI-Based Level Detection and Optimisation of Assistive Robot Maneuverability book will be useful for postgraduate students as well as final year undergraduate students researching on robotics area especially using the latest python software with focus on artificial intelligence techniques.

Siti Fauziah Toha
Department of Mechatronics,
Kulliyyah of Engineering,
International Islamic University Malaysia.
2021

ACKNOWLEDGEMENT

We would like to acknowledge the help from the following colleagues, Muhammad Rafeeq, Zakariya Zainol, Aisyah Ibrahim, for their unprecedented support and valuable suggestions. This book would not have been complete without the moral support and prayers of our family and friends. Last but not least, the Malaysian Ministry of Higher Education (MOHE) supported this work under the project grant: SRCG20-046-0046 IoT-Based Visually Impaired Community (VIC) Geospatial Tracking with Swarming RoVision (SR), RM20,000. 23 Disember 2020 - 22 Disember 2022. (Collaborative research between IIUM - UiTM - UMP - Southampton Univesity Malaysia).

CONTENT

PREFACE	iii
ACKNOWLEDGEMENT.....	iv
LIST OF FIGURES	vii
LIST OF TABLES	ix
 Chapter 1: Introduction	 1
1.1 Overview.....	1
1.2 Problem Statement	2
 Chapter 2: System Overview.....	 3
2.1 Recent Development of Assistive Device.....	3
2.2 Autonomous Staircase Detection Robot	4
2.3 Artificial Neural Network Controller (ANN)	6
2.4 Image Processing in Autonomous Robot.....	13
 Chapter 3: System Design	 20
3.1 Overview of Rovision System	21
3.2 Performance Metrics.....	22
3.3 Hardware	24
 Chapter 4: System Development.....	 29
4.1 Collecting Training and Validation Data	29
4.2 Python software programming	36
4.3 Experiment Setup	44
 Chapter 5: Results and Analysis	 46
5.1 Overview.....	46
5.2 Results and Analysis.....	46
5.3 Discussion	56

Chapter 6: Conclusion & Future Work	57
6.1 Conclusion	57
6.2 Future Work	57
REFERENCE	58
ABOUT THE AUTHORS	60

LIST OF FIGURES

Figure 2.1: Method from Previous Researcher Using Tri-Star Wheel Robot (Thu et al., 2019)	4
Figure 2.2: Design of The UGV and The Result of Edge Detection from Video Feedback (Masood et al., 2017)	5
Figure 2.3: Example of Artificial Neural Network System With 2 Hidden Layers	7
Figure 2.4: Artificial Neural Network System based on research (Medina-Santiago et al., 2014)	8
Figure 2.5: Illustration on the path of Forward and Backward Propagation	11
Figure 2.6: Example of application of Object Identification	14
Figure 2.7: Block diagram for Convolutional Neural Network Form Article (Sumit, 2018)	15
Figure 2.8: Block Diagram process in Single Shot Detection (SSD) from (Forson, 2017)	16
Figure 2.9: Comparison of Object Detection Models (Forson, 2017)	17
Figure 2.10: Result obtain in research (Wolff, 2017)	18
Figure 2.11: Block diagram converting TensorFlow to TensorFlow Lite provide in (Alake, 2020)	19
Figure 3.1: Flowchart for Rovision System Design	20
Figure 3.2: Rovision V1.0 Design	21
Figure 3.3: Flowchart of Rovision Operating System	23
Figure 3.4: Circuit for Rovision	24
Figure 4.1: Infographic of Object Detection Workflow	29

Figure 4.2: Images for Dataset.....	30
Figure 4.3: Interface for FastStone Software.....	32
Figure 4.4: List of XML file.....	33
Figure 4.5: Labelling Image Process	33
Figure 4.6: CSV File Coordinate of the Label Image	34
Figure 4.7: List of TensorFlow Version	35
Figure 4.8: List of Command to Create A Virtual Environment.....	36
Figure 4.9: Training Process using Anaconda Prompt	37
Figure 4.10: Coding for Adding Line to the Bounding Box and Frame.....	38
Figure 4.11: Motor.py and Ultrasonic.py Python Script.....	38
Figure 4.12: Experiment Venue.....	39
Figure 5.1: Example SSD Structure with 6 Convolution Layers.....	41
Figure 5.2: Graph of Error Minimisation	42
Figure 5.3: Result of Detecting Descending Staircase After Training the Model	42
Figure 5.4: Result of Detecting Ascending Staircase After Training the Model	44
Figure 5.5: Result as Revision Closed to Ascending Staircase	44
Figure 5.6: Result as Revision Detect Descending Staircase.....	47
Figure 5.7: Result as Revision Closed to Ascending Staircase.....	48
Figure 5.8: Result as Detecting Descending Staircase.....	48
Figure 5.9: Result as Revision Detect Descending Staircase.....	49

LIST OF TABLES

Table 2.1: Illustrate the Result from Different Staircases (Masood et al., 2017).....	6
Table 2.2: Illustrate the Type of Activation Function for Neural Network (Feng & Lu, 2019)	9
Table 2.3: Notation Variable for Related Equation Above Based on Research (Medina-Santiago et al., 2014)	12
Table 3.1: Raspberry Pi Specification	25
Table 3.2: Camera Specification	25
Table 3.3: Ultrasonic Sensor Specification	26
Table 3.4: DC Motor Specification	26
Table 3.5: Motor Driver Specification	27
Table 3.6: Lithium-ion Battery Specification	27
Table 3.7: Buck Converter Specification	28
Table 5.1: The Result for Ultrasonic Sensor Accuracy Test	47
Table 5.2: Total Loss Value Every 1k Steps	49
Table 5.3: The Accuracy of Ascending Staircase	51
Table 5.4: The Accuracy of Descending Staircase	52

Chapter 1: Introduction

1.1 Overview

Statistically, based on the research that was made by World Health Organization, there are about 285 million visually impaired people worldwide in 2012. Unlike us, they have difficulty to live their normal life. A normal and easy task such as to move from one place to another safely or to differentiate an object from another may be an easy for us but not for them as these people had lost one of their 5th senses which is their sight. Thus, they can only depend on the other senses which are hearing, touch, smell and taste. However, for them, to depend on the other four senses still cannot ensure their safety whenever they want to do activities. Thus, a lot of devices has been built and developed for them to ease their mobility.

Rovision is one of the robots that has been developed by Dr Siti Fauziah and her team for the purpose of helping the visually impaired people to detect obstacle. Its functions can merely be the same as the traditional guide dog as this robot contains suitable sensors and actuator to aid the disable. Rovision was built with several parts which are robot body, white cane holder, wheels and robot head. Multiple sensors are attached to the robot's head and body. Sensors are one of the most important part in Rovision as it allows the robot to collect data from robot's surrounding area. Data that has been collected by multiple sensors will be filtered out to get the final path for the robot to go and successfully avoid the obstacle.

Controller is one of the crucial components need to have in every autonomous mobile robot. It acts as the brain for the robot in determining the action of the robot, how it moves and the way robot processing the data. Controller is a set of code that has been written in the microprocessor with multiple of conditions and algorithms in order to make the robot act according to desired goal. The common microcontroller that usually use to create an autonomous robot are Arduino and Raspberry Pi. Multiple versions of Integrated Development Environment (IDE) are available to simulate and test the efficiency of the controller. Arduino IDE, Python IDE, MATLAB are the most popular IDE when it comes to design and implement the controller to a robot.

Selecting suitable controller for the robot is very important depending on the situation and the surrounding of the robot. There are a lot of controller available in mobile robot field such as sensor filter, fuzzy logic, artificial neural network and etc. Implementing the controller to a robot will make the robot able to follow the desirable output. All the controller

is based on mathematical algorithm which can be change depending on the condition. Most of autonomous mobile robot only implement one controller for a robot but sometimes controller can be combined in order to get more precise and accurate output.

1.2 Problem Statement

Visually impaired people often faced difficulty to undergo their daily life naturally due to limited movement capability, and proper guiding device. Without hundred hours of training on the route, it is difficult for them to move safely. Although reports have shown some improvements using global positioning system (GPS) receiver with braille access to localise information in any outdoor environment, it still does not guarantee a hazard free journey without the visually able person's help. However even an independent visually impaired person walking alone is susceptible to danger. Therefore, the need to model an algorithm for a localised swarm-like cooperative and communicator which could be used among the visually impaired is crucial. This would be useful for the targeted group to enable them to walk in a larger group with assurance that they know their company during the travel and also for safety precautions.

The biggest challenges are to help the visually impaired people to navigate through the ascending and descending staircase. Visually impaired people mostly felt down due to they are not aware with the present of the staircase. They also have difficulty in finding the staircase whenever they in building, surrounding and unfamiliar areas. Choosing the right sensors and controller is crucial as the robot need to be the 5th sense for this people.

In this book, a raspberry pi is showcases to be used to combine multiple signals from camera sensors and 4 ultrasonic sensors into one clear data that can be guide the robot in order to execute the next action such as move forward or reverse whenever staircase is detected. Thus, a control system is very much important element to ensure the Rovision 'thinking' ability what to be done for the next action.

Chapter 2: System Overview

2.1 Recent Development of Assistive Device

Assistive devices are commonly used where people mostly depended on the device to accomplish their tasks and works. For blind and visually impaired people (VIP), the development of technology really gives them benefits as now they can be independent without needed any helps from other people. Nada et al., 2015 reported that almost 85% of the people with visually impaired experienced tripping when walking and thus resulting in a fall. Thus, assistive device helps visually impaired people to know their surrounding accurately and help them to avoid obstacles and holes on the ground compared to traditional method which used long cane and guide dog.

There are varieties of designs has been made by previous researcher to create a fully functional assistive device for the impaired people. All the design usually used different kind of sensors and sometimes mix few sensors to obtain stable and accurate data. (Mahmood et al., 2015) Sonar Assistive Device for Visually Impaired people is one of the projects that has been done by previous researcher. Sonar technology is used to help those visually impaired people to self-walking and give confident to them. As a result, it is found that 90 % of users confident to use the device to help them detecting obstacles.

Smart stick also been developed in order to improve the efficiency of normal long cane. In this device, developer added sensors such as ultrasonic sensors and water sensor that can detect high-level and low-level obstacle and hole specifically water puddle. Based on the result of the research, blind and visually impaired people feel much safer after few attempts using smart stick as the average speed of the blind user increase respect to number of attempts. (Nada et al., 2015)

This book showcases the research undertaken in order to improve efficiency of the designed assistive device (Rovision) for the blind, specifically in detecting staircase. The vital problems related with maneuverability of the VIP is during stair ascending and descending which is prone to tipping over and falling. The current designed assistive device is only compatible in certain and in limited environment only, therefore with the outcome of this research on the use of assistive device is widen.

2.2 Autonomous Staircase Detection Robot

Most of previous technology for blind and visually impaired people assistive device do not have a system that helps user to detect the present of staircase or any incline surface. this problem should be taken for consideration as most of the terrain are not smooth and have many obstacles. The device most probably assume staircase as an obstacle that need to be avoided. Thus, a optimize control system needs to be embedded into this technology to distinguish between staircase and obstacles so that it can increase the effectiveness and efficiency of the technology.

This advance technology is famous and usually implement in autonomous mobile robot that required robot to be able to detect and climb object like staircase. Earthquake Rescue Robot is one of the examples of successful product in implementing this technology. Different kind of approached has been done by previous researcher in creating an autonomous staircase detection robot such as legged by (Łabęcki et al., 2011), wheel by (Thu et al., 2019) and track by (Masood et al., 2017), (Adiwahono et al., 2014), (Mihankhah et al., 2009), (Roumeliotis et al., 2002). This variety of method creates many ways to solve the problem and different output.

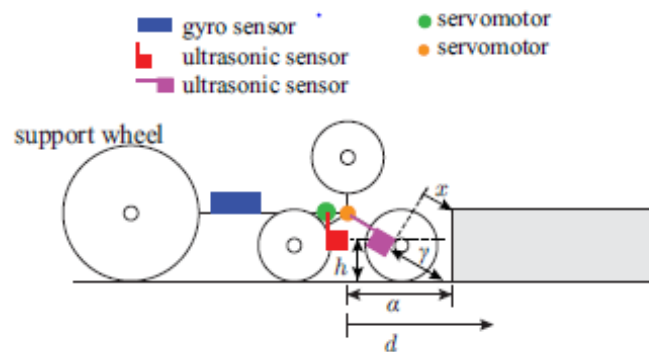


Figure 2.1: Method from Previous Researcher Using Tri-Star Wheel Robot (Thu et al., 2019)

In research done by (Thu et al., 2019), the robot as shown in Figure 2.1 uses two ultrasonic sensors as the medium to detect the obstacle, gyro to detect the position of the robot and two encoders to measure the rolling of tri-star wheels. Based on the results,

positioning of the sensors will affect the output data from the robot. Ultrasonic sensor is mounted at the bottom of the robot also at one of the tri-star wheel as sensor to detect the distances from obstacle and the distance from ground respectively. As the robot start to climb the stair, ultrasonic sensor and gyro start to detect changes in value and position. Thus, the result from this research shows that by combining the data from all the sensors will result more accurate output data.

Another paper from (Masood et al., 2017) take different approach in creating stair detection robot as shown in Figure 2.2. The author mentioned that the purpose of creating this robot is for urban search and rescue purposes. Image processing method has been used in this robot in order to detect the orientation and the position of the stair effectively. It also used the video feedback to avoid any collision with object by using the edge detection of the stairs. An algorithm was developed on MATLAB to find the required angle to climb the stair. The algorithm is important as it determines the position of the different steps by comparing different intensity of light from video feedback.



Figure 2.2: Design of The UGV and The Result of Edge Detection from Video Feedback (Masood et al., 2017)

Table 2.1: Illustrate the Result from Different Staircases (Masood et al., 2017)

Sr. No	Actual	Calculated			Error (%)
		Iteration 1	Iteration 2	Iteration 3	
1	15	13.25	13.9	14.19	-8.13
2	22	20.33	21.4	20.8	-5.26
3	27	25.15	26.22	25.7	-4.85
4	31	31.8	32.7	32.4	4.19
5	35	37.3	38.1	36.4	6.48

Table 2.1 shows the result of few iterations of slope from five different staircases. The result was gained from the video feedback after edge detection and noise cancelation process taken by build in camera on the robot. As the error is so small, the data still can be used to flip the robot arm to exact angle in order to climb the stair. Researcher come up with variety of method in developing a robot that able to detect staircase and object. Thus, in this book, an artificial neural network is discussed as a controller to the robot system to navigate the robot in the right direction that clear from any obstacles.

2.3 Artificial Neural Network Controller (ANN)

2.3.1 Overview

In general, an Artificial Neural Network is a computing system that mimic the works of human brain in processing the data and making the right decision. The self-learning in the system makes it compatible to solve problem that hard for human and it also led to the development of Artificial Intelligence. Processing units comprise ANNs which consist of inputs and outputs in turn. The inputs are what the ANN learns to generate the desired output from. an input, an output and a hidden layer are 3 layers that must have to construct an artificial neural network as shown in Figure 2.3. Hidden layer is a layer exist between input and output that helps the system to reach the desired output based on the given

input. Hidden layer can be single or multiple layers based on the complexity given to the system.

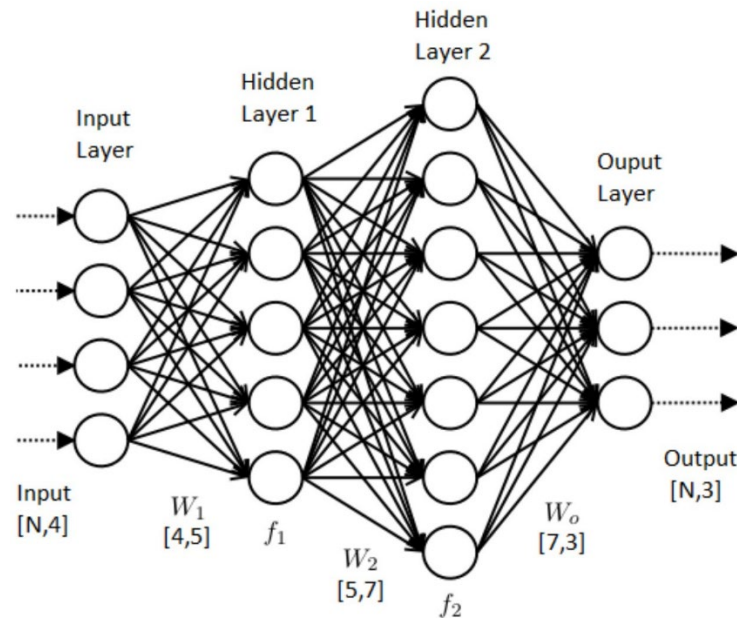


Figure 2.3: Artificial Neural Network System With 2 Hidden Layers (Toha and Tokhi, 2008)

A study has been done by (Medina-Santiago et al., 2014) on autonomous object avoidance mobile robot based on artificial neural network mention that navigate a robot in unknown environment is one of the most crucial challenges in mobile robot field as those robots have to be able to evade the difficulties they face on their way to a goal. Thus, neural network is one of the best solutions in solving problem of navigation vehicles due to its ability to learn nonlinear relationship between the output and the input data.

For the Rovision controller, three ultrasonic sensors were embedded in front of the mobile robot as the medium to detect the present of the obstacles. The data sensed by the sensors then will be the input for the neural network system. The output for the system is the actions that robot will execute based on specific condition. The condition is a logic in term of mathematical algorithm that generate by the hidden layer referring to the input data given by the sensors. All set of input and output are in form of vector. 0 in output data represent action that machine will ignore and 1 is action that will be execute. 4 set of actions were set in the output layer which are turn left, turn right, recoil and advance. The neural network for the system can be represent as Figure 2.4.

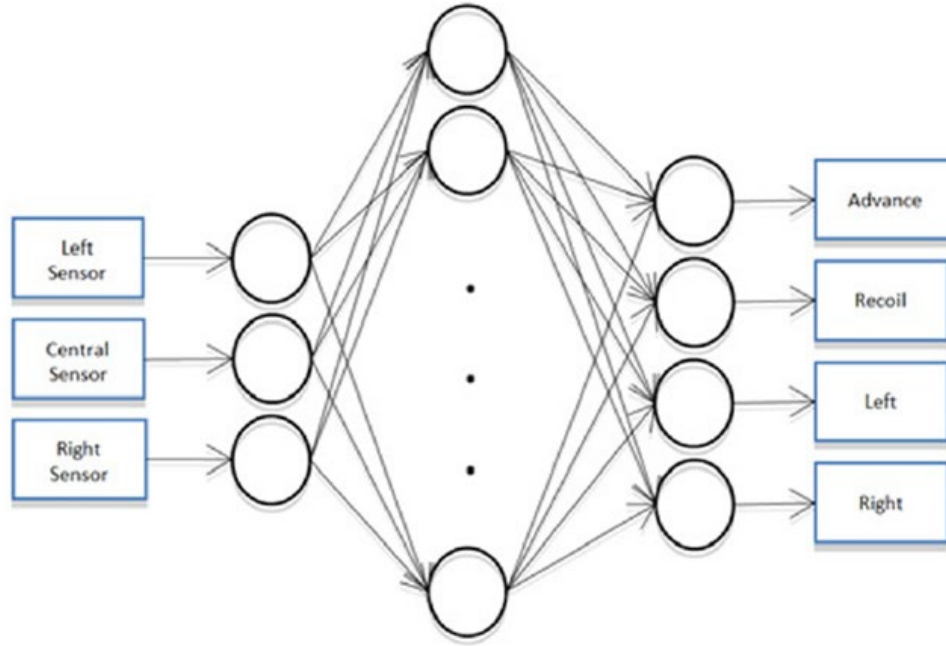


Figure 2.4: Artificial Neural Network System based on research (Medina-Santiago et al., 2014)

2.3.2 Artificial Neural Network Weight (Parameter)

The efficiency of the artificial neural network is depending on the weight between the nodes. A weight represents the strength of the connection between units. If the weight between node 1 and 2 has greater magnitude, it means that neuron 1 has greater influence over neuron 2. A weight decreases the importance of the input value. Based on the research (Medina-Santiago et al., 2014), the calculation for the weight between the input and the first hidden layer was represented in equation (1). All the data and bias weights are stored in the memory of the embedded system in matrix form.

$$n_j^o = \sum_{i=1}^q W_{ji}^o P_i + b_j^o \quad (2.1)$$

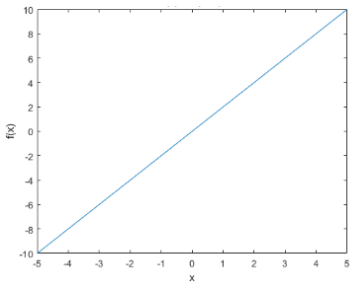
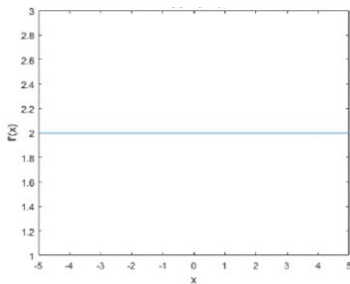
where W_{ji}^o is the weight connecting the input layer to the hidden layer, P_i is the component of the input vector, and b_j^o is the gain of the hidden layer, extra weights in the networks

2.3.3 Activation Function

Neural network activation functions are a crucial element of deep learning. Activation functions enhance the efficiency of a deep learning model, its accuracy, and also the computational efficiency of training a model which can make or break a large-scale neural network. Based on the study (Ertuğrul, 2018), he mentions that activation function helps the neural network controller to set the limit boundary. It prevents the output signal generate from $-\infty$ to $+\infty$. Without activation function, the relationship between input and output is linear regardless number of hidden layers applied in the system.

Numeric data points, called inputs, are fed into the neurons within the input layer in a neural network. Every neuron has a weight, and multiplying the input number by the weight gives the neuron output to the next layer. The activation function is acting like a logic gate between the input that feeds the current neuron, and the output that goes to the next layer. The simplest activation function can be a step function which can turn output on and off. There are 2 types of activation function which are the linear and nonlinear activation function. Research has been made by (Feng & Lu, 2019) on various activation function and based on the research, linear function has the exact equation as straight line while nonlinear activation function has different type of shapes which indicates that with nonlinear activation function, robot can adapt with variety data and to tell the different between the output.

Table 2.2: Type of Activation Function for Neural Network (Feng & Lu, 2019)

Type of Graph	Curve of Function	Curve of Derivate
Linear		

Nonlinear

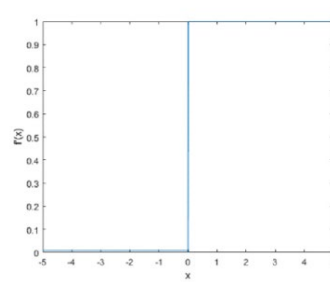
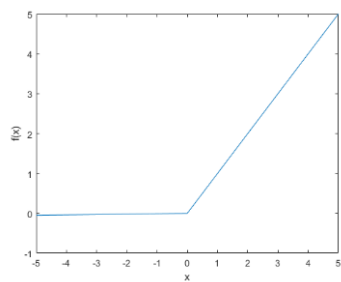
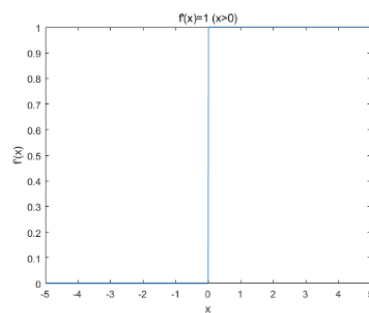
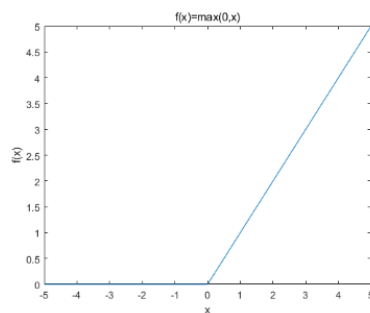
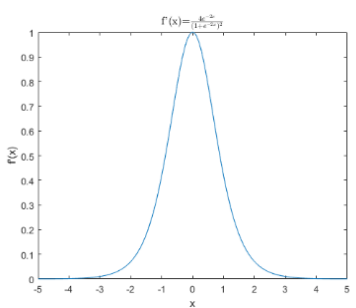
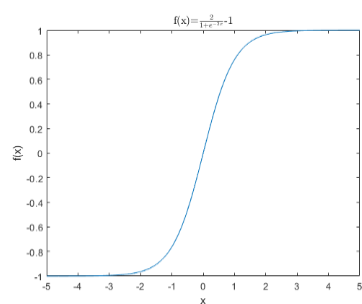
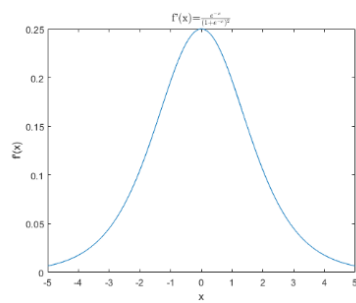
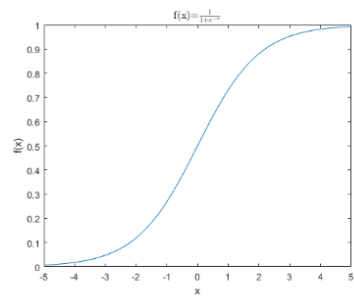


Table 2.2 shows some of the common types of activation function in neural network. Sigmoid, Tanh, Rectified Linear Unit (ReLU) and Leaky ReLU are the name of the nonlinear activation function respectively. Note that the different between the derivative of ReLU and Leaky ReLU is ReLU has 0 input at the negative side while Leaky ReLU is close to 0 usually 0.01 or so at the negative side (Feng & Lu, 2019).

2.3.4 Forward and Backward Propagation

Propagation according to Merriam-Webster brings the meaning of spread or extension. The term propagate is used in neural network as it illustrates the movement of the data starting from the input to the output. Forward Propagation is when the data from the input pass through all the nodes in the hidden layers to output layers, while Backward Propagation doing the inverse movement from forward propagation.

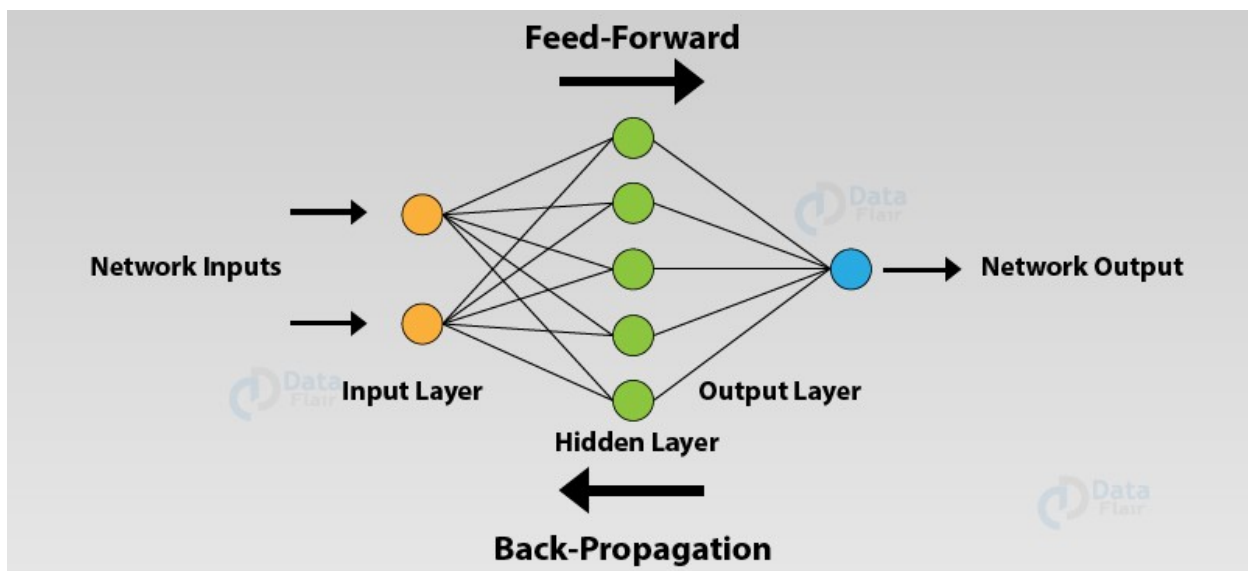


Figure 2.5: Feedforward and Backward Propagation for Neural Network Layers (Toha and Tokhi, 2008)

Forward propagation method basically a process that make the controller calculate the error in the system by finding the difference between the gain output and the desired output. Assuming the construction of the neural network similar as Figure 2.5 which consist 3 layers. The general equation is taken from the article by (Nagasawa et al., 2005).

$$z^{(2)} = XW^{(1)} \quad (2.2)$$

$$a^{(2)} = f(z^{(2)}) \quad (2.3)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (2.4)$$

$$\hat{y} = f(z^{(3)}) \quad (2.5)$$

Table 2.3: Notation Variable for Related Equation Above Based on Research (Medina-Santiago et al., 2014)

Code Symbol	Math Symbol	Definition	Dimensions
X	X	Input Data, each row in an example	(numExamples, inputLayerSize)
y	y	target data	(numExamples, outputLayerSize)
W1	$W^{(1)}$	Layer 1 weights	(inputLayerSize, hiddenLayerSize)
W2	$W^{(2)}$	Layer 2 weights	(hiddenLayerSize, outputLayerSize)
z2	$z^{(2)}$	Layer 2 activation	(numExamples, hiddenLayerSize)
a2	$a^{(2)}$	Layer 2 activity	(numExamples, hiddenLayerSize)
z3	$z^{(3)}$	Layer 3 activation	(numExamples, outputLayerSize)

Back propagation method differs to forward propagation as it starts at the output layer where it takes the error calculated from forward propagation and feed the input back to the input layer. Referring to (Farooq et al., 2010), the back propagation method uses the gradient algorithm which can make the value of mean- squared error decrease to it minimum. At the moment error is minimize, the weight value become stable. The weight equation can be represented as:

$$W_{n+1} = W_n + \eta \sum_p \delta_{ij} x_i \quad (2.6)$$

where W_{n+1}^0 is the new weight connecting the input layer to the hidden layer, x_i is the component of the input vector, and δ_{ij}^0 is the gain of the hidden layer, extra weights in the networks

Hence, forward propagation or feed forward method is used to calculated the error from the output between the target and experiment output while for back propagation method, it is used to reduce the error produce in feed forward method to stabilize the neural system

2.4 Image Processing in Autonomous Robot

2.4.1 Overview

Image processing is part of the artificial neural network that has been design in order for the computer to identify specific object. Image processing field is wide where researcher applied the image processing method on many applications. (Galvez et al., 2019) mentioned that image processing is functional in solving many problems as the machine can have the vision like human and can identify things accurately. The process in the image processing required high computing power as the system need to be train with a lot of pictures. With the help of latest technology like high performance GPU and CPU, training a data set be much easier.

The method of image processing can be different based purpose using the image processing. Based on research by (Tydén & Olsson, 2020), the processing of digital images can be categorized into image enhancement, image restoration, image analysis, and image compression. In image enhancement, an image is changed and manipulated so that a human can extract useful data from it. Image restoration is where corrupted images is processed from which there is a statistical or mathematical description of the degradation so that it can be reverted. Image analysis techniques allow an image to be processed so that information can be extracted from it automatically.

2.4.2 Object Identification/ Recognition

Object identification or recognition is one of the latest technologies invented using image processing method. This advance technology able to label object on the image or live feed correctly by creating boarder around the object. One of the machines that use this technology is autonomous driving car. The implementation of this image processing makes the machine work effectively as it can recognize all the object in front of it through camera. The more advance the technology used on the autonomous car, the more reliable the car can be to the user as shown in Figure 2.6.

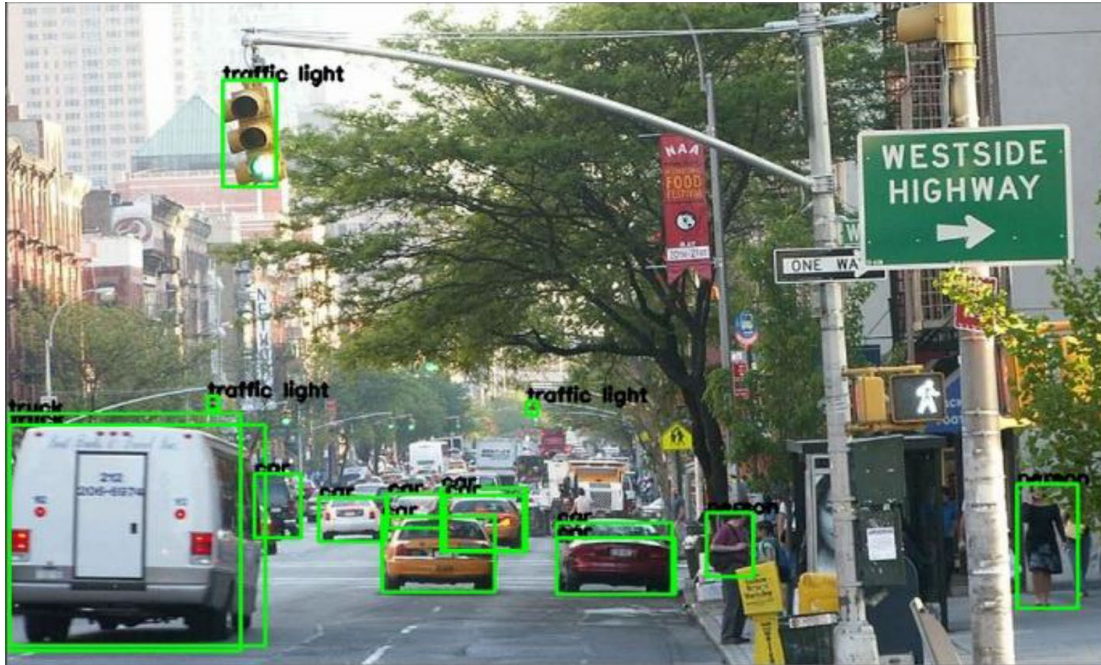


Figure 2.6: Application of Object Identification (Galvez et al., 2019)

Based on the research paper by (Galvez et al., 2019), he mentioned that there are several approaches made by previous researchers on detecting and tracking an object using video feed. Extracting information from the video is a very difficult as it required complex algorithm that make the machine be able to track and detect a moving object smoothly. Creating a boarder and colored mask on the object are two methods that available for object tracking and identification. These two methods might be the same but have a different result. (Galvez et al., 2019) explained that creating a boarder on the object might be the best among the two because it can capture the whole part of the object with less noise and the machine has no problem on creating and updating the boarder on a moving object on the video or live feed. Creating mask on the object has the same purposes with the first one but it will create a silhouette based on the object. The disadvantages of this method are it cannot cover all the object. This problem can be solved by improving the camera to have more resolution

2.4.3 Convolution Neural Network

CNN is a well-known method used for computer vision and over the past few decades it has improved in accuracy. An input layer, an output layer, and intermediate hidden layers

are built up into a CNN. Figure 2.7 provides an overview of the architecture of a convolutional neural network.

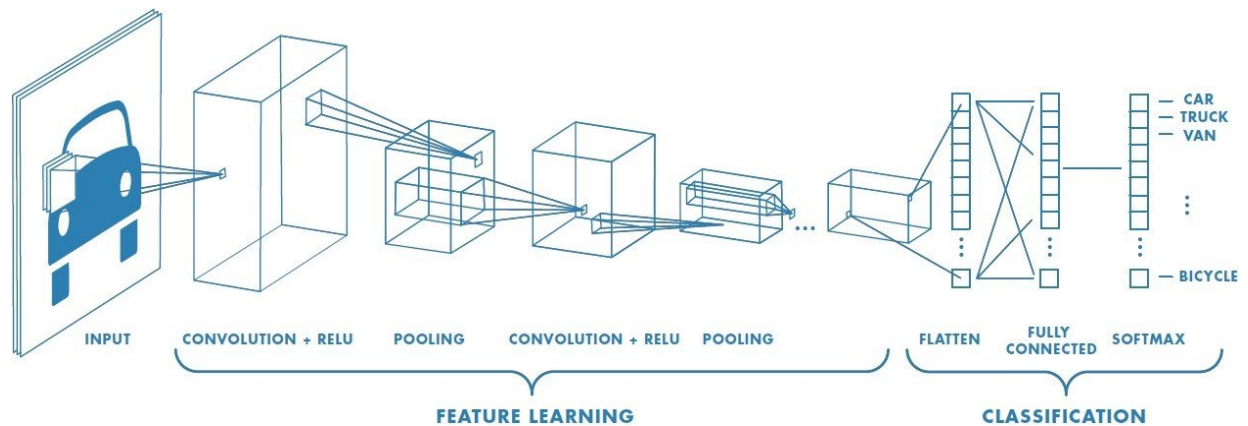


Figure 2.7: Block diagram for Convolutional Neural Network (Sumit, 2018)

Based on (Sumit, 2018), Convolutional neural network mostly used to do image recognition, image classification, object and face detection. CNN is a special type of neural network architecture that designed to work with two-dimensional image data. The name convolutional is given as it applies convolution method in the convolutional layer. Within the setting of a CNN, a convolution could be a straight operation that includes the duplication of a set of weights with the input, much like a conventional neural arrange. Given that the procedure was designed for two-dimensional input, the multiplication is performed between a cluster of input information and a two-dimensional cluster of weights, called a filter. This orderly application of the same filter over an image may be an effective thought. In case the filter is outlined to detect a particular sort of highlight within the input, at that point the application of that filter efficiently over the complete input image allows the filter an opportunity to find that highlight anyplace within the image.

Feature learning and classification are two process that involve in CNN. Feature learning is process where the input data's information will be filter out using kernel in every pixel on the image. The process will include the pooling process and the application of activation function. In classification process, the data then will go through the traditional neural network where forward propagation and backpropagation process occur. This process will determine the class of the image based on data extraction from feature learning process. Thus, CNN is one of the best architectures that can be used in deep

learning and machine learning as its capability to operate multiple layers in processing information in an image.

2.4.4 Single Shot Detection (SSD)

Single shot multiple box detection is one of the architectures that commonly used in convolutional neural network. There are different types of algorithm available in convolutional neural network for instance YOLO (You Only Look Once), Faster R-CNN, R-CNN and etc. the existence of multiple architecture is to adapt with latest technology and convolutional neural network can be apply at any devices. Faster R-CNN and R-CNN are commonly applied on a desktop which have high processing power while YOLO and SSD are created to make the system applicable to mobile devices such as raspberry pi and smartphone.

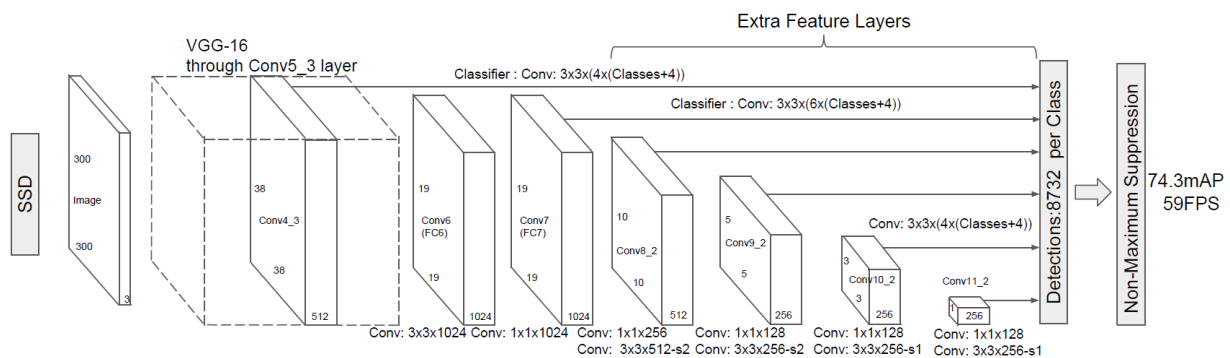


Figure 2.8: Block Diagram process in Single Shot Detection (SSD) (Forson, 2017)

Apart from the capability to operate in mobile devices, SSD also able to detect multiple boxes in single shot. Based on article written by (Forson, 2017), he mentions that SSD was designed for detecting multiple categories. The structure of the model uses multiple scale convolution bounding box outputs attached to the network. This feature is very useful in lots of situation where the camera needs to identify multiple objects as shown in Figure 2.8.

A comparison on accuracy between all primary object detection methods was done by PASCAL VOC 2017 using coco datasets based on article from (Forson, 2017). Based on Figure 2.9, the accuracy performance by SSD methods is on the second highest before R-CNN which mean that SSD is the most suitable and accurate method to apply in mobile devices like this project.

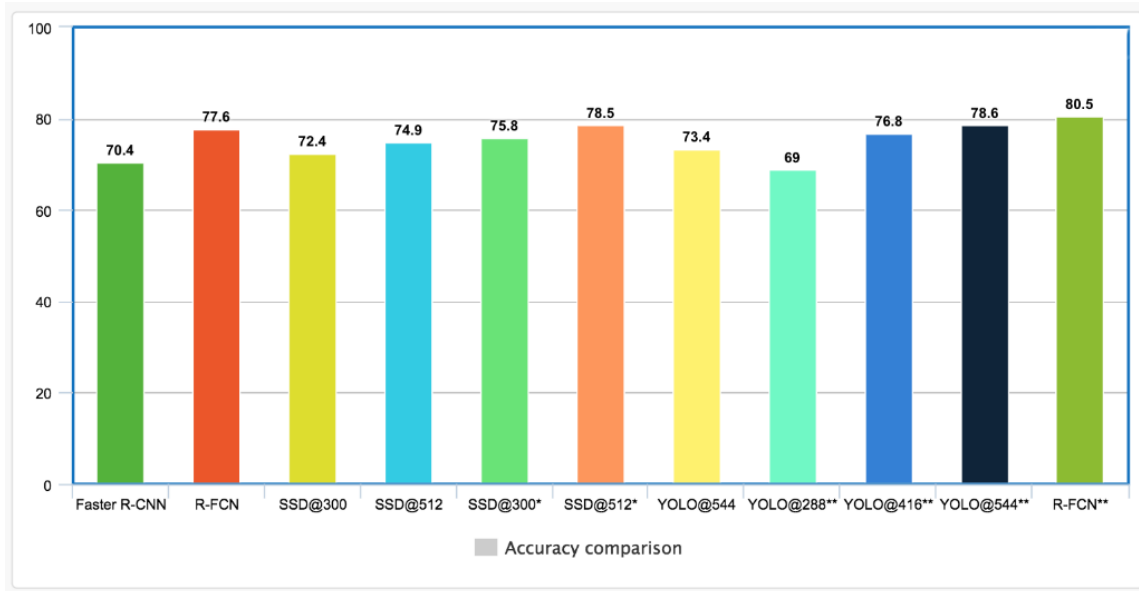


Figure 2.9: Comparison of Object Detection Models (Forson, 2017)

2.4.5 MobileNet Inception in Single Shot Detection

Single shot detection is selected to be used in this project due to its capability to detect multiple objects at once. In TensorFlow, there are two available type of Convolution neural networks available for SSD algorithm that capable to run in low computing mobile devices which are the SSD Mobilenet V2 and SSD Inception V2. Both methods use different type of convolution neural network which standard CNN used in Inception and Depthwise CNN used in Mobilenet. A previous researcher (Wolff, 2017) has made comparison on both architectures through their performance in term of accuracy, training time, prediction time and model size. Based on the result obtained from the research shown in Figure 2.10, (Wolff, 2017) conclude that the performance of the Mobilenet was better compared to the Inception in all aspects.

The accuracy of Mobilenet model increase up to 72.5% from Inception. The result also illustrate that the Mobilenet perform well compared to the Inception in every test done by the researcher such as prediction time (950ms for Mobilenet, 5000ms for Inception) and file size (900KB for Mobilenet, 84MB for Inception). Those comparison is important as the architecture will be applied on the low computing mobile device. The result below illustrates three types of model that can be apply in machine learning. But we only focus on the Mobilenet and Inception model as both of the architectures are available in the TensorFlow framework. Thus, SSD mobilenet V2 was choosen to be the architecture model in this project due to its excellent performance and adaptability on the mobile devices compared to the other model available in TensorFlow.

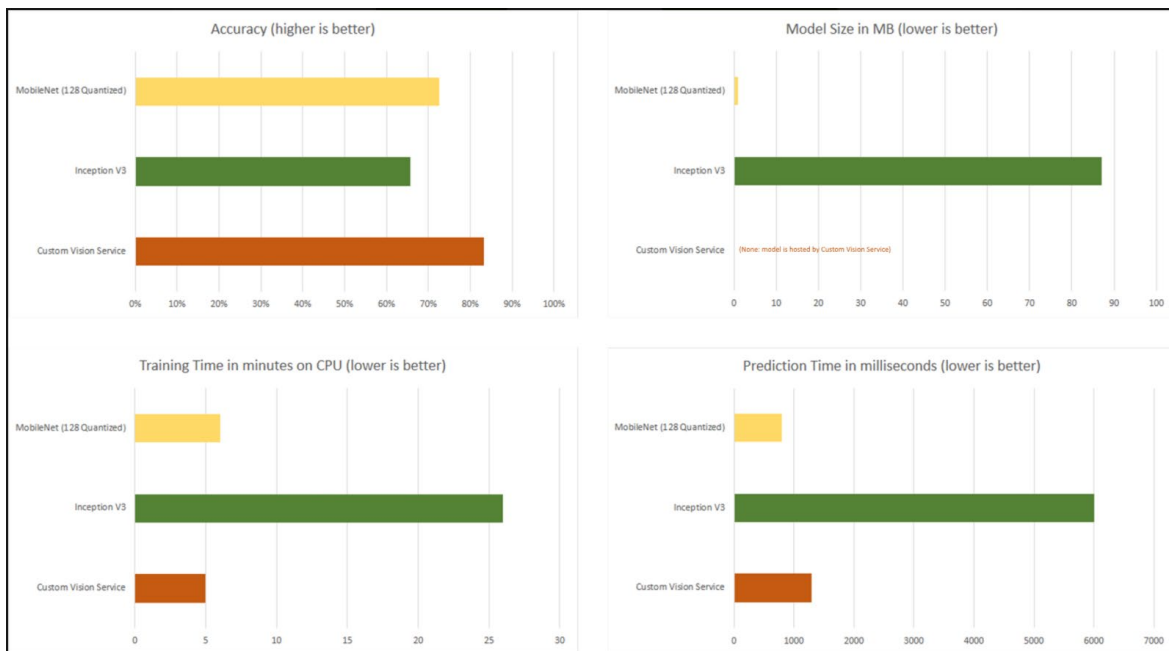


Figure 2.10: Result obtain in research (Wolff, 2017)

2.4.6 TensorFlow VS TensorFlow Lite

Referring to official TensorFlow website, TensorFlow is an end-to-end opensource framework. It has a robust, scalable ecosystem of software, libraries and community services that helps researchers to quickly create and deploy machine learning driven applications to push the state-of-the-art in machine learning and developers. TensorFlow

offers a series of instruments that allow the implementation of a machine learning model for a variety of purposes and environments.

As stated in the article by (Alake, 2020), TensorFlow Lite takes existing TensorFlow models and transforms them in the form of a tflite file into an optimized and efficient version as shown in Figure 2.11. The streamlined model is small enough to be stored on computers and sufficiently precise for effective inference to be carried out. The advantages of TensorFlow Lite are it can operate more efficient on mobile devices; it allows machine learning to run quickly with low latency without using external API or server which mean it can be run online.

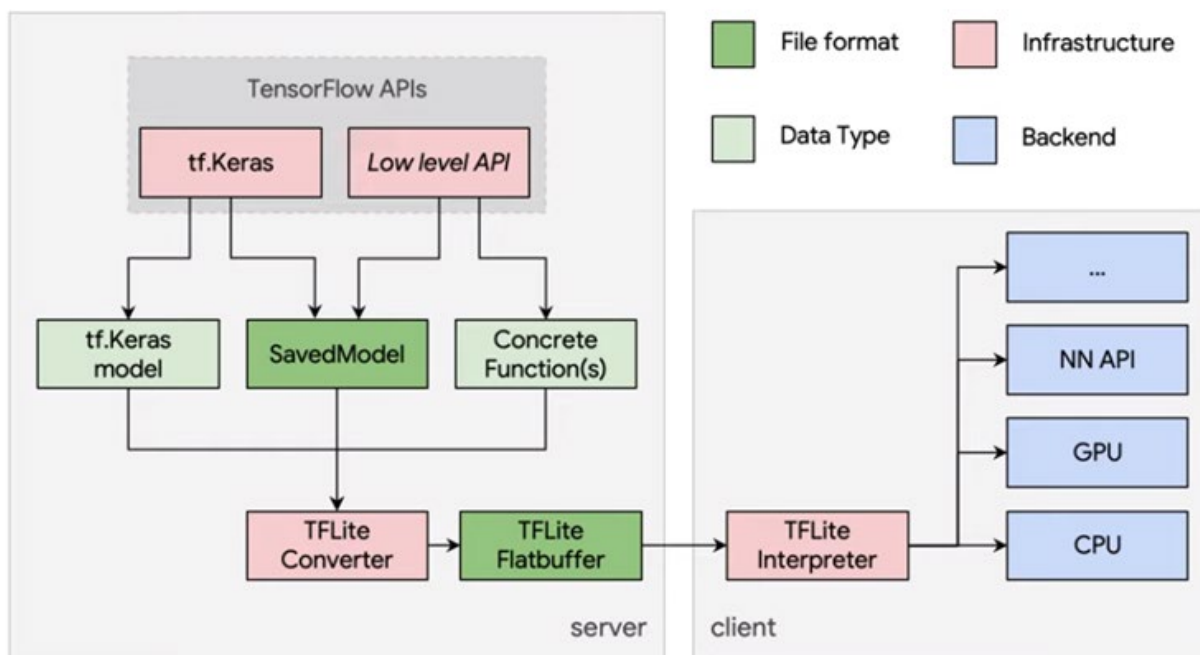


Figure 2.11: Block diagram converting TensorFlow to TensorFlow Lite provide in (Alake, 2020)

Chapter 3: System Design

The Rovision system design contains three major parts which are modelling of system, controller design and hardware implementation. This chapter will explain methods based on a flowchart that determines all the activities that has to be done. Figure 3.1 shows the overall Rovision system design

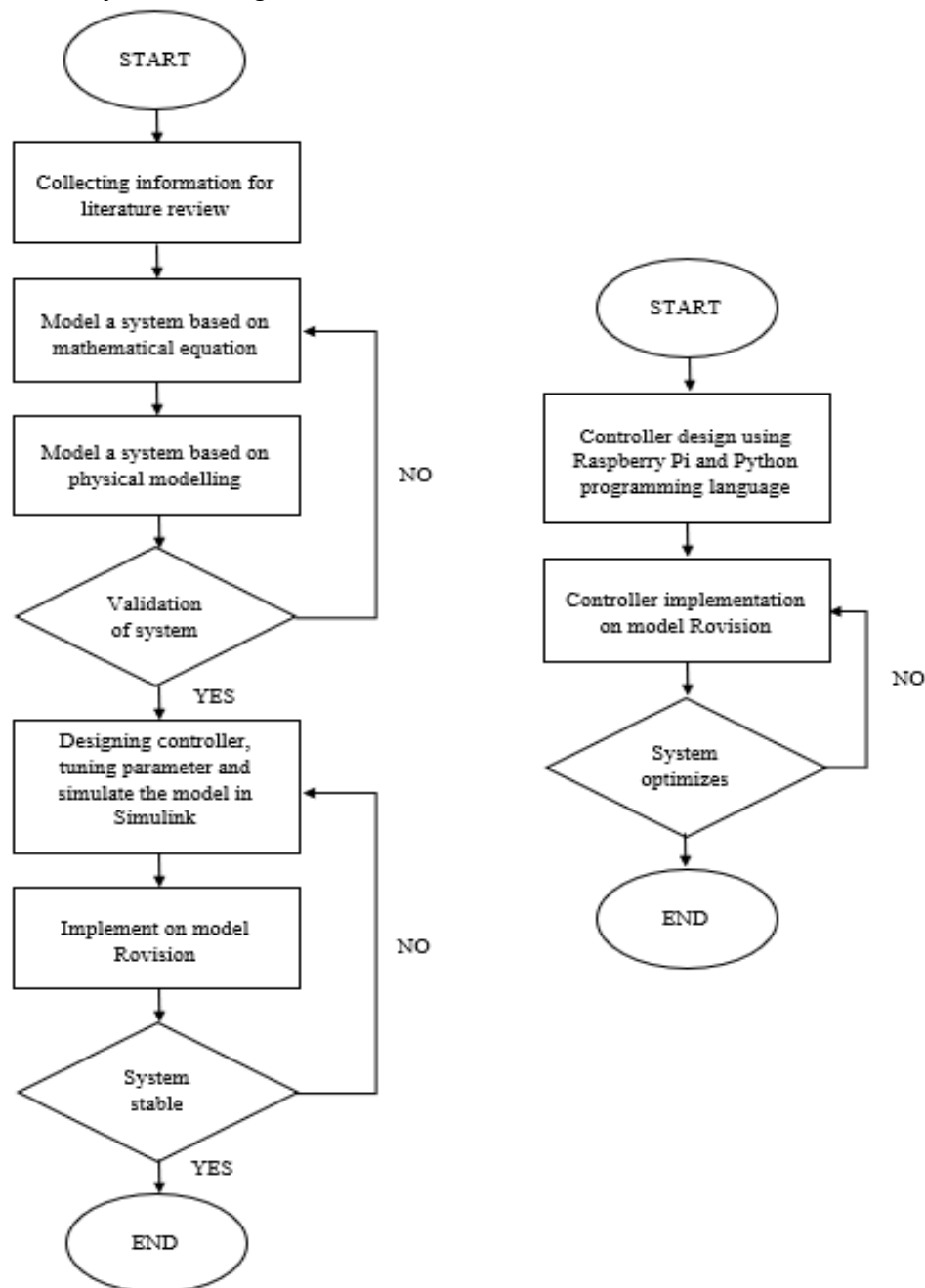


Figure 3.1: Rovision System Design Flowchart

3.1 Overview of Rovision System

Rovision is an autonomous assistive robot for blind and visually impaired people to help them move around. The robot was created to navigate user around and avoid obstacles to prevent any injuries to the user. Multiple sensors attached on the robot's head and it also have speed controller to vary the speed of the motor whenever the sensors detected any obstacles. Rovision V1.0 consist of a sigle camera and four ultrasonic sensors that are attached on the robot as its main sensors as shown in Figure 3.2.

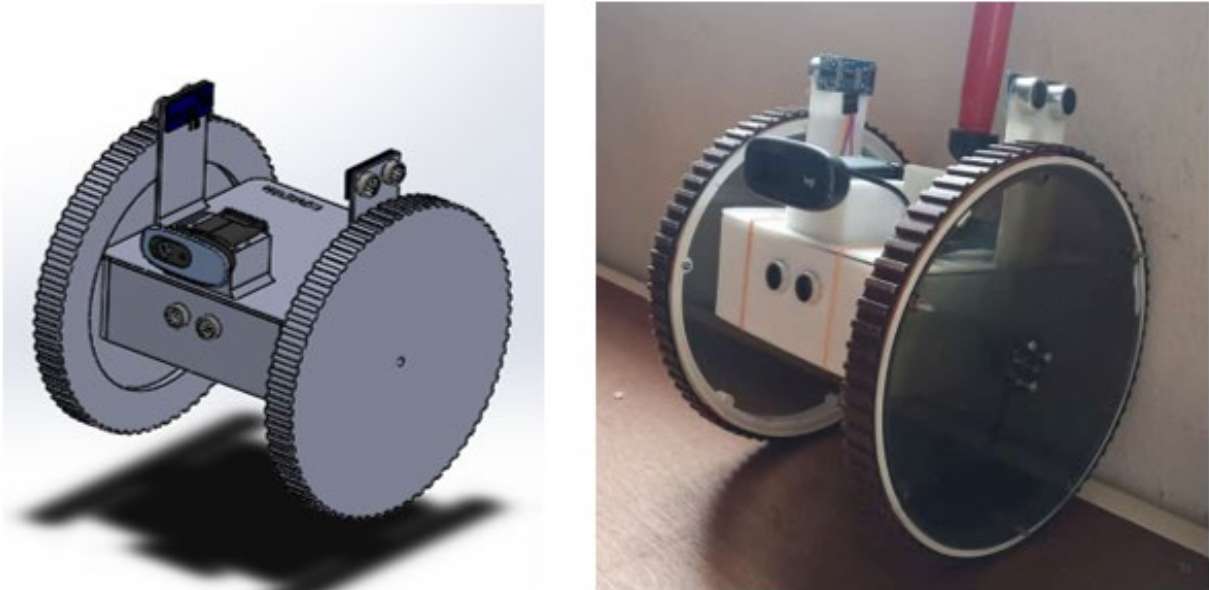


Figure 3.2: Rovision V1.0 System Design

Results from system simulation shows that artificial neural network has successfully provides the ability to avoid the wall and obstacle smoothly. With the improvement of the microcontroller, Raspberry Pi offers much more convincing result compared to Arduino microcontroller as the Raspberry Pi has the capability to save data which is necessary when running an artificial neural network system. One of the drawbacks of using Arduino microcontroller is it needs repetitive training as the data will be erased every time the Arduino is turned off. While Raspberry Pi is one of the cheapest computing devices that available in the market. It mostly used as a portable computer for mini projects because it has its own microprocessor. Using Raspberry Pi for Rovision, it will help the robot to produce constant results and most importantly it only requires one-time training.

A camera is used as the main sensors along with four other ultrasonic sensors are connected to the Rovision prototype. A camera will give a vision to the machine to make it be able to “see” the surrounding. Image processing method is used on the Rovision in order to make the robot be able to detect a staircase. Image processing will aid the robot in term of object clarification and localization. The system can be set to detect anything and classify the object into specific classes. As of a focus of this book, image processing is trained only to detect the ascending and descending staircase. Further alteration of the system can be made by adding another class for desired object.

An additional ultrasonic sensor is used as the edge detector sensor. The purpose of this sensor is to add more safety to the robot. This sensor is programmed to detect any changes of distance on the floor in order to warn about the descending stair in front. Figure 3.3 shows the overall flowchart of Rovision operating system. This sensor is not connected with the other three sensors which mean that it will not use artificial neural network system to activate.

3.2 Performance Metrics

The important points involved with the performance metrics in analysing the system performance are discussed based on the context of this experiment:

True Positive (TP): System detect stair when stair is available

False Positive (FP): system detect stair but no stair available

True Negative (TN): system not detect stair when no stair available

False Negative (FN): system not detect stair but stair is available

Accuracy is a calculation that indicates whether a model is being trained correctly and how it performs. Accuracy is calculated using the following formula:

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (3.1)$$

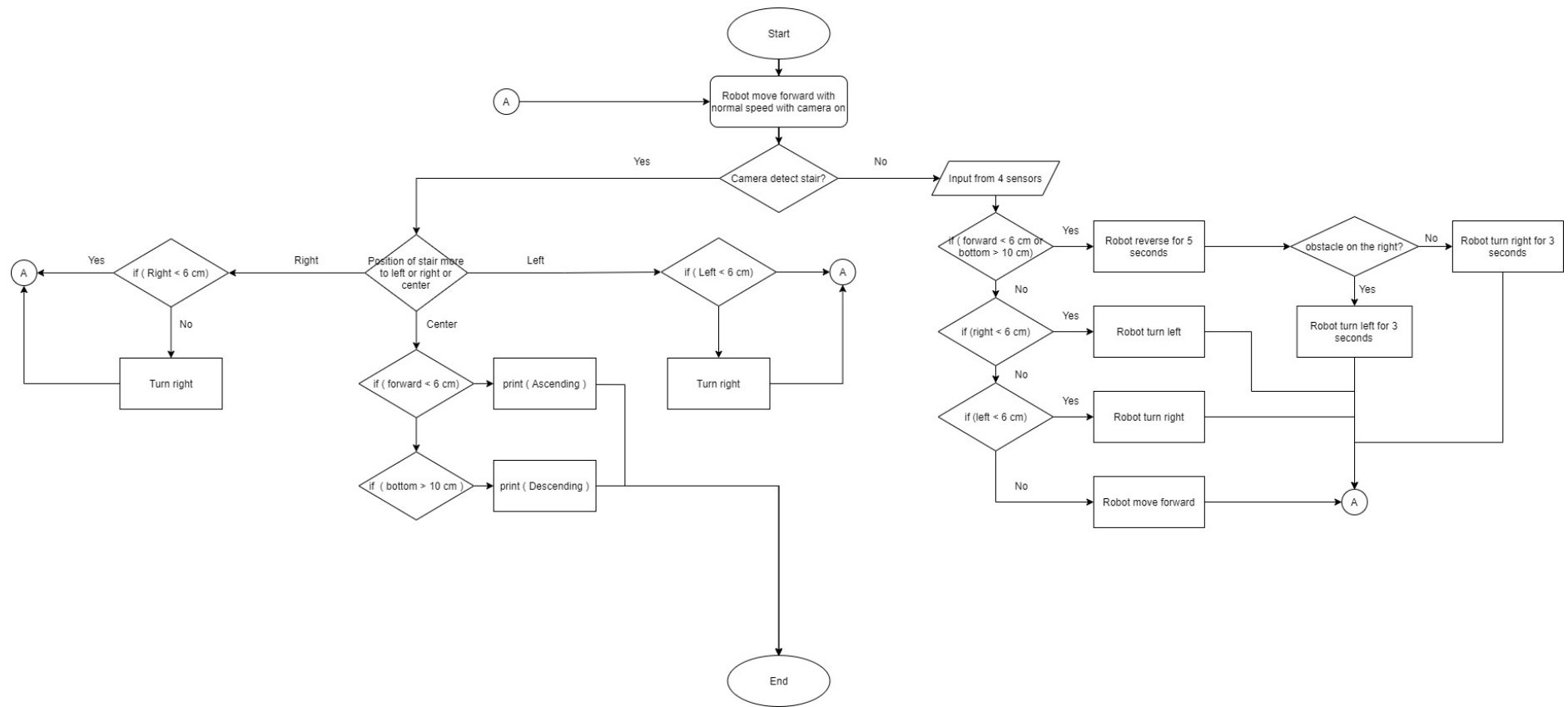


Figure 3.3: Flowchart of Rovision Operating System

3.3 Hardware

The hardware is successfully developed as discussed in Section 3.1. The Rovision circuit diagram is shown in Figure 3.4 comprises of Raspberry pi, Webcamera, Ultrasonic sensors, DC motor, Motor driver and Battery:

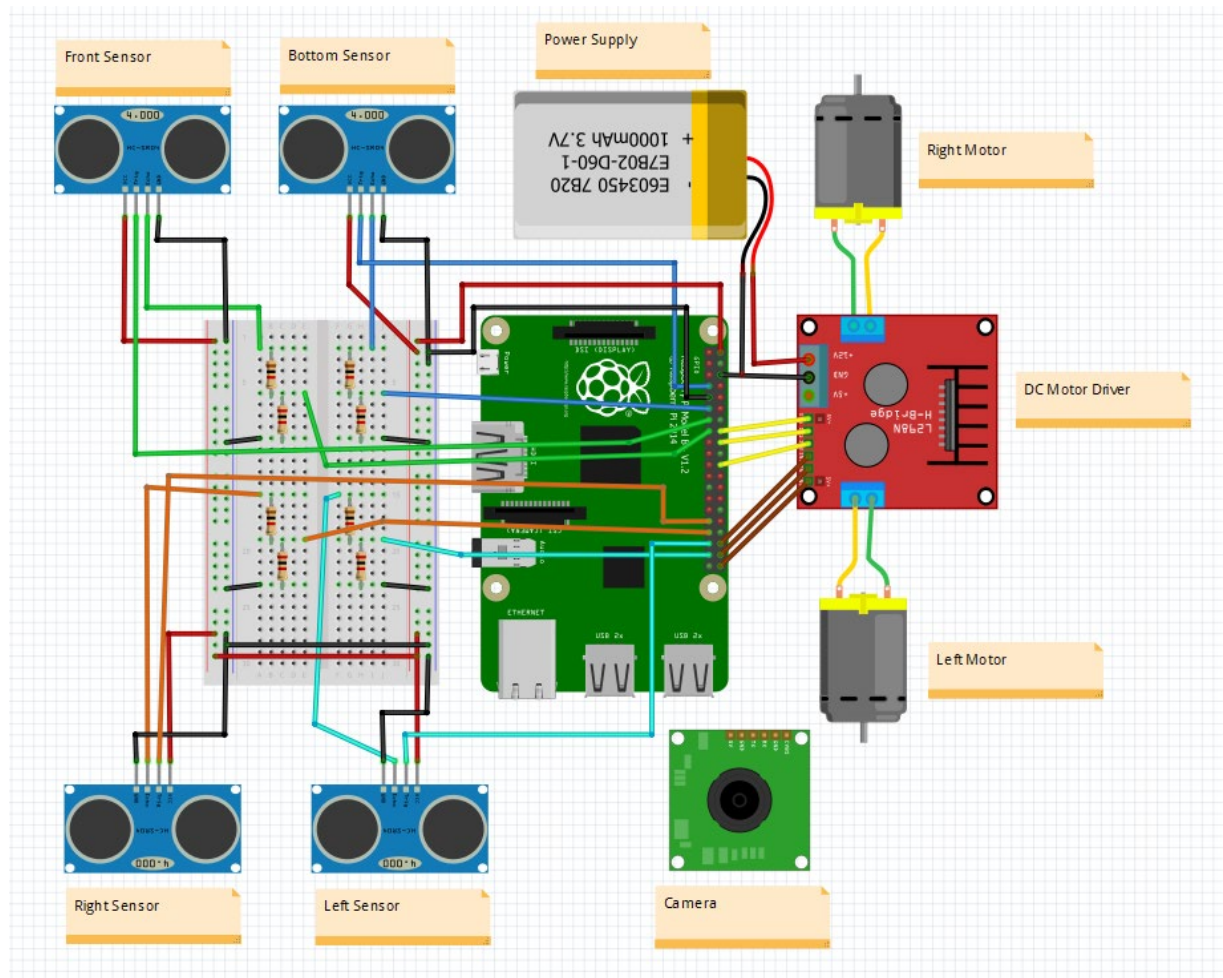



Figure 3.4: Circuit for Rovision


The selection of hardware is depending on the purpose of the project. All the component are selected based on the capability on executing the targeted output. In developing an autonomous mobile robot. The component must be portable and can operate in dynamic situation. The list of components used in this project are presented in the next section.

Table 3.1: Raspberry Pi Specification

Component	Specification	
 RASPBERRY PI 4 MODEL B	CPU	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
	RAM	4GB LPDDR4-3200 SDRAM
	Pin	Raspberry Pi standard 40 pin GPIO header
	WIFI/BT	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
	Price	RM 239.00 (quantity = 1)

The Raspberry pi specifically RP 4 model B is a microprocessor that can act as a tiny computer which suitable to any project that required small computation system and portable as tabulated in Table 3.1. The raspberry pi 4 used the latest processor that can withstand more processing work compared to the previous version of raspberry pi.

Table 3.2: Camera Specification

Component	Specification	
 LOGITECH C270 WEBCAM	Weight	75 g
	Resolution	720p/30fps
	Focus	Fixed focus
	Build-in	Microphone (Mono)
	Price	RM 172.58 (quantity = 1)

The camera is the main sensor in this project due to the objective to locate the position of the stair. This sensor has the best quality image which can create video on 30fps. Frame per second (FPS) is important as it can affect the smoothness of the robot movement as tabulated in Table 3.2.

Table 3.3: Ultrasonic Sensor Specification

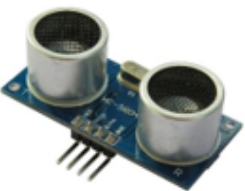

Component	Specification	
 <p>ULTRASONIC SENSOR HC-SR04</p>	Weight	8.5 g
	Operating voltage	5V
	Operating current	<15mA
	Range	2cm to 80cm
	Price	RM 4.00 (quantity = 4)

Table 3.4: DC Motor Specification

Component	Specification	
 <p>SPG30E-60K DC MOTOR</p>	Weight	175 g
	Operating voltage	12V
	Operating current	0.41A
	RPM	75 RPM
	Torque	2.6 kg-cm
	Price	RM 68.00 (quantity = 2)

For ultrasonic sensor as tabulated in Table 3.3, it become the second sensor for the robot. The application of ultrasonic sensor is to perform object detection when detecting other than stair. Precision is key factor for the robot to operate better. In this project, 4 sensors are used, namely, front sensor, bottom sensor, left sensor and right sensor.

For the selection of motor as tabulated in Table 3.4, DC motor is used to maneuver the robot to the desired location which is toward the staircase. Thus, it doesn't have any requirement to select the suitable motor.

Table 3.5: Motor Driver Specification

Component	Specification	
 L298N DUAL H-BRIDGE MOTOR DRIVER	Weight	26 g
	Drive voltage	5V – 35V
	Drive current	2A
	Price	RM 7.00 (quantity = 1)

Table 3.6: Lithium-ion Battery Specification

Component	Specification	
 18650 LI ION 3.7V BATTERY	Weight	48 g
	Nominal Voltage	3.7V
	Nominal Capacity	2850mAh
	Price	RM 4.00 (quantity = 3)

Motor driver as tabulated in Table 3.5 is used in this project as the motor shield for the DC motor. The purpose of using this component is to control the speed of the motor by regulating the voltage supply to the dc motor and to be able to change the direction of motor either clockwise or anti-clockwise.

Battery used to power up the microprocessor such as raspberry pi as tabulated in Table 3.6. For this project, the robot required 12V to power the motor driver for the dc motor to move. 3 lithium-ion battery are used to fulfil the requirement by supplying 11.1V.

Table 3.7: Buck Converter Specification

Component	Specification	
 BUCK CONVERTER	Weight	20 g
	Nominal Voltage	3.7V
	Nominal Capacity	2850mAh
	Price	RM 15.00 (quantity = 1)

Buck converter is a device that capable to step down the voltage from power supply to supply to specific components or devices as tabulated in Table 3.7. In Revision, buck converter is important as it used to step down voltage supply from battery which is 11.1V to 5V required to power up Raspberry pi. The target is to power up Revision using only 1 power supply which is the lithium-ion battery.

Thus, all the components are important in making the Revision robot be able to move. Notice that in the circuit, voltage divider is used between the ultrasonic sensor and raspberry pi. This is due to the capability of the raspberry pi to read the input from the ultrasonic sensor. Raspberry pi can only read signal from 3.3V only while the ultrasonic sensor's echo send signal in form of 5V. using voltage divider, the voltage can be dropped to desired voltage for the raspberry pi.

Chapter 4: System Development

4.1 Collecting Training and Validation Data

The illustration shown in Figure 4.1 projected the steps in making an object detection system that can detect specific object on an image, a video and a live video using raspberry pi. The process includes collecting related images, labelling image, creating custom pre-trained models in TensorFlow, converting file to TensorFlow-lite, implement the custom pre-trained model to raspberry pi and executed the program using USB camera. Further elaboration on each step will be on next sub section.

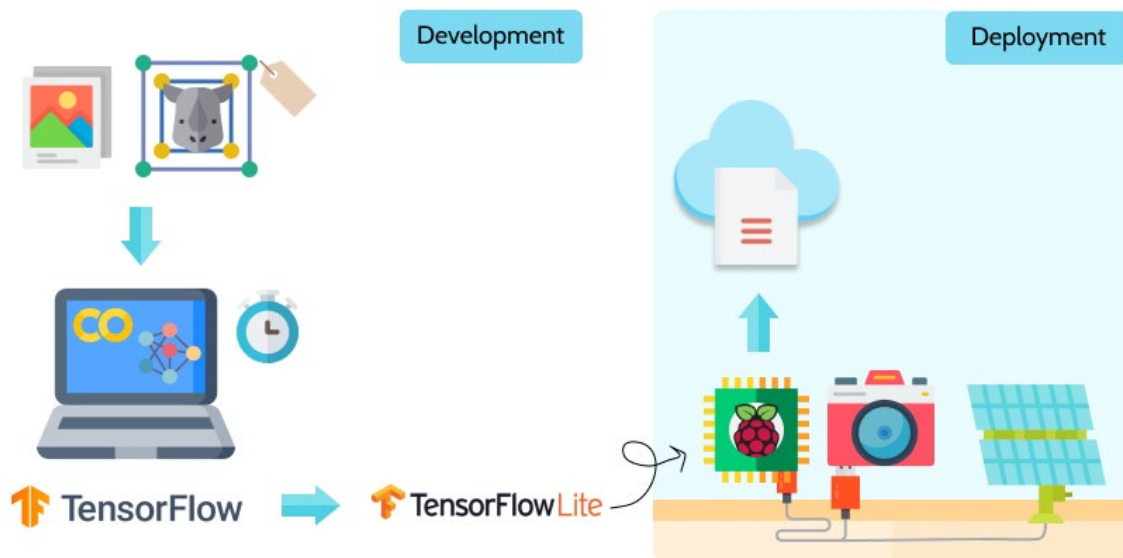


Figure 4.1: Infographic of Object Detection Workflow

4.1.1 Collecting Image for Training and Test Data

To create a reliable system, where the system can detect the object precisely, a set of training data and test data need to be collected to train the machine to recognize the object. In this project, 1000 pictures of staircase, ascending and descending conditions are used and be divided into 2 files, the train file and the test file. 20% out of 1000 pictures are used as the test image while the rest are used for the train image. The picture must

be taken in different kind of situations such as low light and different angles to create variety of information for the system to learn about the staircase as shown in Figure 4.2.

The dimension of the training and test picture must below 720x1280 and the picture data must not exceed 200KB. It is to ensure the computer can train the picture smoothly and take short time to complete the training. For this project, at the beginning of training the picture, the average of the picture size taken by phone camera is around 2MB to 4MB which already exceed the necessary size and the dimension for the picture is 2160 x 4560. Using a free software called FastStone Photo Resizer 4.3, the software can resize all the pictures for the project to below 100KB with dimension of 227 x 480 as shown in Figure 4.3.

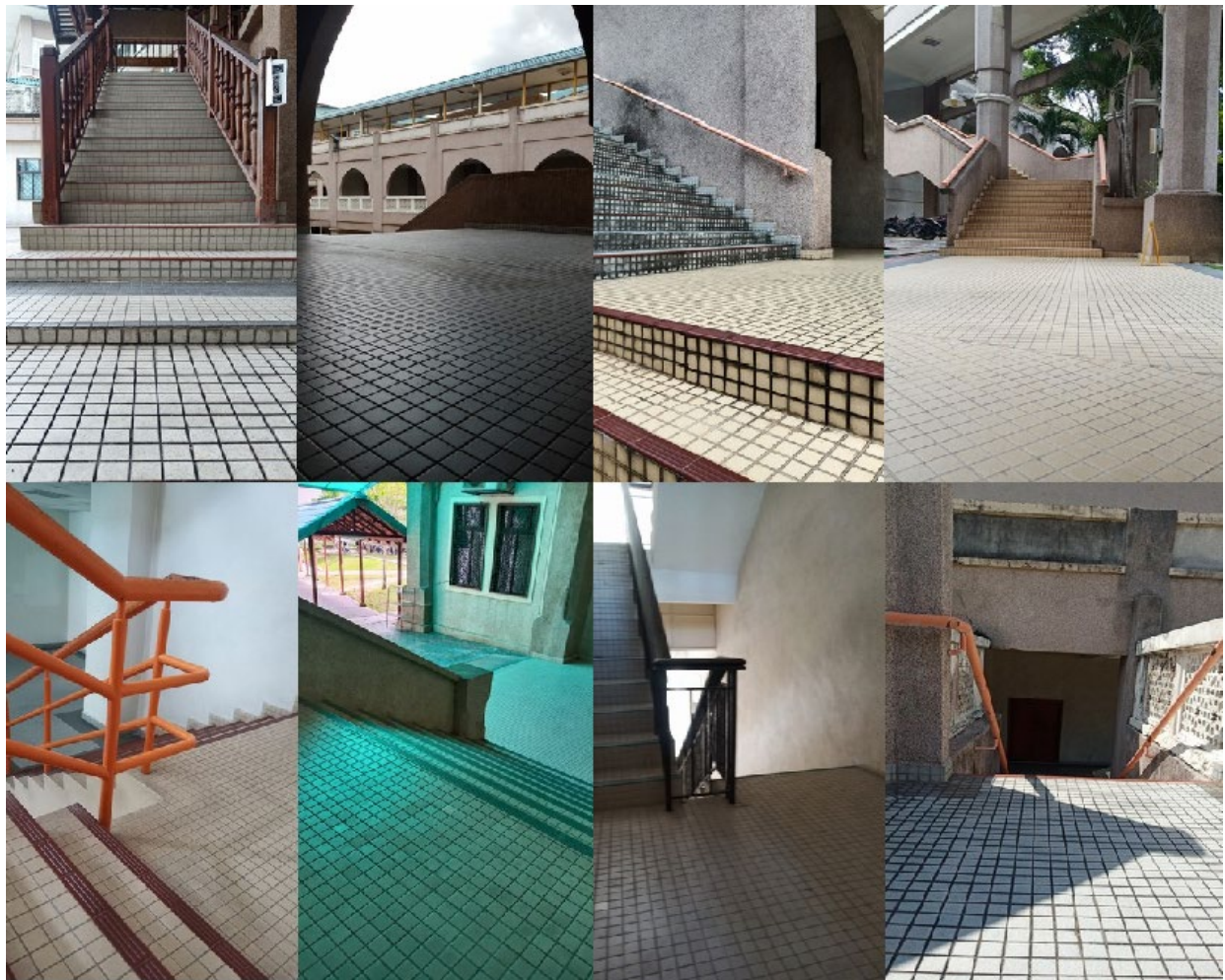


Figure 4.2: Some Ascending and Descending Image in the Dataset

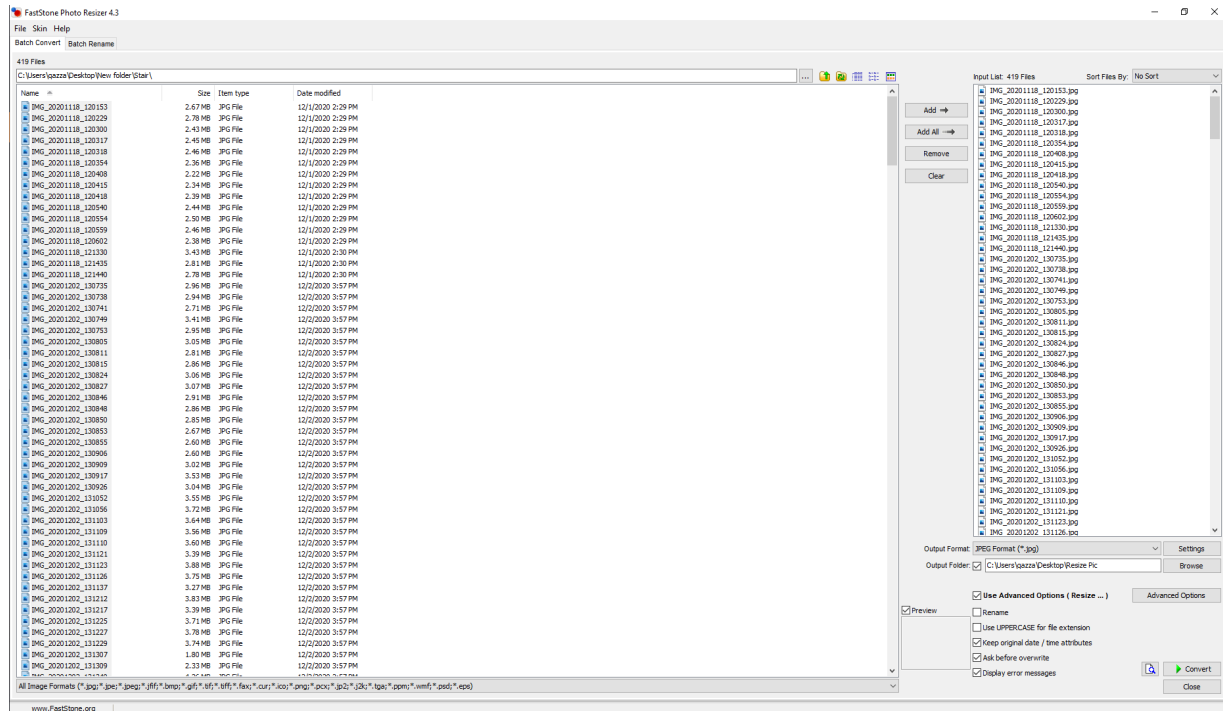


Figure 4.3: Interface for FastStone Software

4.1.2 Labelling Image

After collecting the train and test image for image processing, labelling image is the next necessary step to create a file that contains information for the system. Labelling image is a process where the desired object in the picture is labeled and grouped into several groups. In this case, the project only needs two categories which are ascending and descending staircase. For all the pictures used in the train and test files, it is compulsory to label each one of the pictures to tell the machine how the object looks like.

This process will make the machine able to identify the position of the targeted object. The labelling image process in Figure 4.4 will save as .xml file that can be referred in Figure 4.5 containing the label data for each image where it then can be converted to .csv files for the next step before the training session starts. Figure 4.6 shows the CSV File Coordinate of the label image.

IMG_20201118_1201...	12/2/2020 6:20 PM	XML Document	1 KB
IMG_20201118_1202...	12/2/2020 6:21 PM	XML Document	1 KB
IMG_20201118_1203...	12/2/2020 6:22 PM	XML Document	1 KB
IMG_20201118_1203...	12/2/2020 6:22 PM	XML Document	1 KB
IMG_20201118_1203...	12/2/2020 6:29 PM	XML Document	1 KB
IMG_20201118_1203...	12/2/2020 6:30 PM	XML Document	1 KB
IMG_20201118_1204...	12/2/2020 6:30 PM	XML Document	1 KB
IMG_20201118_1204...	12/2/2020 6:30 PM	XML Document	1 KB
IMG_20201118_1204...	12/2/2020 6:30 PM	XML Document	1 KB
IMG_20201118_1205...	12/2/2020 6:31 PM	XML Document	1 KB
IMG_20201118_1205...	12/2/2020 6:31 PM	XML Document	1 KB
IMG_20201118_1214...	12/2/2020 6:31 PM	XML Document	1 KB
IMG_20201202_1307...	12/2/2020 6:35 PM	XML Document	1 KB
IMG_20201202_1307...	12/2/2020 6:36 PM	XML Document	1 KB
IMG_20201202_1307...	12/2/2020 6:36 PM	XML Document	1 KB
IMG_20201202_1307...	12/2/2020 6:38 PM	XML Document	1 KB
IMG_20201202_1307...	12/2/2020 6:38 PM	XML Document	1 KB

Figure 4.4: List of XML file

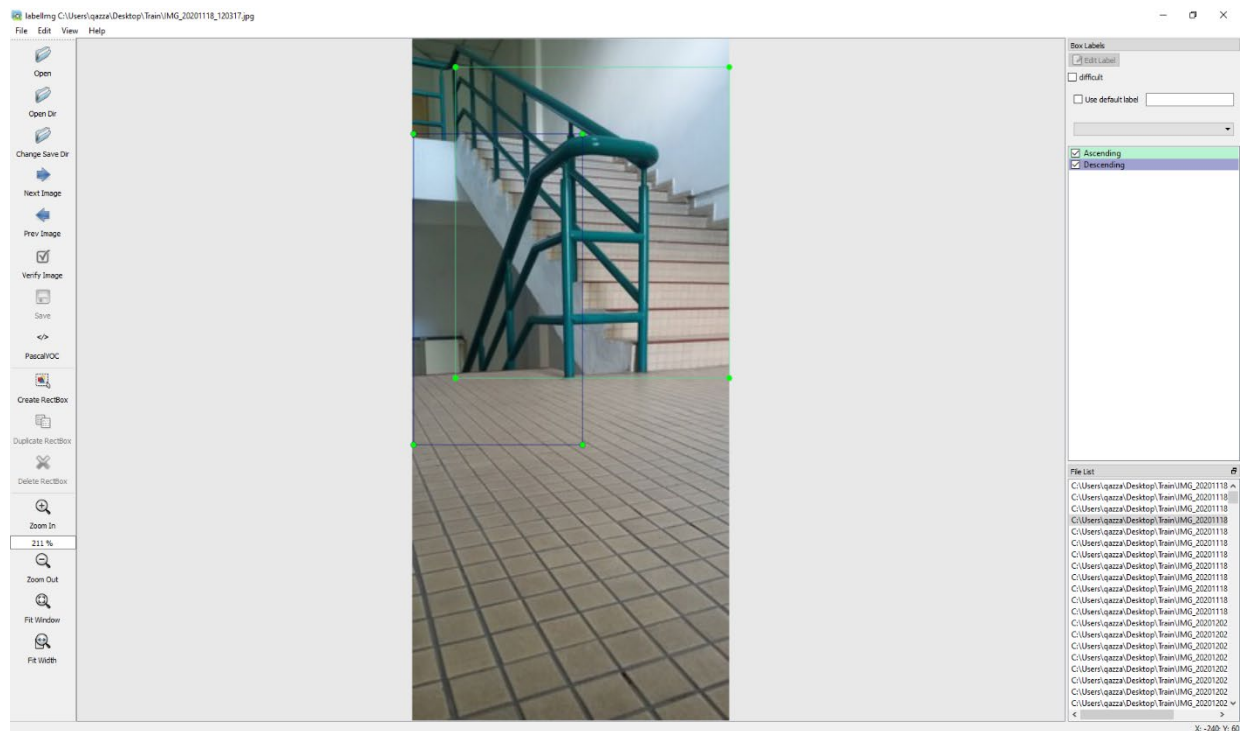


Figure 4.5: Labelling Image Process

filename	width	height	class	xmin	ymin	xmax	ymax
IMG_20201118_120153.jpg	227	480	Descending	27	86	204	263
IMG_20201118_120229.jpg	228	480	Ascending	5	114	228	344
IMG_20201118_120300.jpg	227	480	Ascending	1	20	209	248
IMG_20201118_120317.jpg	227	480	Ascending	31	20	227	239
IMG_20201118_120317.jpg	227	480	Descending	1	67	122	286
IMG_20201118_120318.jpg	227	480	Ascending	28	15	227	239
IMG_20201118_120318.jpg	227	480	Descending	1	76	145	260
IMG_20201118_120354.jpg	227	480	Ascending	18	59	219	268
IMG_20201118_120408.jpg	227	480	Ascending	126	76	227	243
IMG_20201118_120408.jpg	227	480	Descending	1	98	112	263
IMG_20201118_120415.jpg	227	480	Descending	23	102	227	248
IMG_20201118_120418.jpg	227	480	Descending	26	120	212	265

Figure 4.6: CSV File Coordinate of the Label Image

4.1.3 Installation of TensorFlow Framework

To train the neural network, a deep learning library TensorFlow was used along with the TensorFlow Object Detection API, which help to simplifies the process of training models for object detection by using pre-trained models from COCO dataset or custom model which need to be train from scratch. Google Collaboratory can be used to edit and run Collab notebooks as it is hosted online and offers free use of GPU (cloud). Another option is to use the processor from personal computer, either use the CPU or the GPU. GPU is more preferable as it can fasten the process of training compared to CPU. In order to run TensorFlow using GPU, a specific software need to be installed into the computer which is the Nvidia CUDA and Nvidia CUDNN.

The version for each software needs to be correct based on the version of TensorFlow installed to avoid unwanted error. The specific type of GPU must be used in order to ensure the GPU fulfil the requirement for image processing. All the information about the software version and the GPU compatibility can be referred at the official page of Nvidia.0 In this project, GTX 1650 SUPER as main GPU to run the TensorFlow version 1.15 using Nvidia CUDA version 10.0.130 and CUDNN version 7.6.5 as shown in Figure 4.7. It is recommended to follow tutorial during installation as manual installation required specific version to work well.

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow_gpu-2.3.0	3.5-3.8	MSVC 2019	Bazel 3.1.0	7.6	10.1
tensorflow_gpu-2.2.0	3.5-3.8	MSVC 2019	Bazel 2.0.0	7.6	10.1
tensorflow_gpu-2.1.0	3.5-3.7	MSVC 2019	Bazel 0.27.1-0.29.1	7.6	10.1
tensorflow_gpu-2.0.0	3.5-3.7	MSVC 2017	Bazel 0.26.1	7.4	10
tensorflow_gpu-1.15.0	3.5-3.7	MSVC 2017	Bazel 0.26.1	7.4	10
tensorflow_gpu-1.14.0	3.5-3.7	MSVC 2017	Bazel 0.24.1-0.25.2	7.4	10
tensorflow_gpu-1.13.0	3.5-3.7	MSVC 2015 update 3	Bazel 0.19.0-0.21.0	7.4	10
tensorflow_gpu-1.12.0	3.5-3.6	MSVC 2015 update 3	Bazel 0.15.0	7	9

Figure 4.7: List of TensorFlow Version

4.1.4 Anaconda IDE

Anaconda IDE is used as the alternative way to install correct version of TensorFlow in computer. With the help of Anaconda IDE, the TensorFlow and both Nvidia CUDA and Nvidia CUDNN will automatically be installed in an environment created by Anaconda IDE in a correct way using few specific conda command. The advantage of using Anaconda IDE for TensorFlow installation is one computer can have many versions of TensorFlow and will not affect one another.

Anaconda IDE helps creating an environment for every TensorFlow installation. The current version using in this project is Anaconda 2020.07 for Python 3.8. The version of Python is not fixed where in the environment, different version of python also can be used such as for this project, the environment creates for the TensorFlow used Python 3.7 to run the PIP installation.

Figure 4.8 shows the command used to create and activate the TensorFlow in Anaconda environment. Notice that the third line of command is not specifying the version of TensorFlow-GPU used in the environment. It will download and install the latest version of TensorFlow which the latest version for now is TensorFlow 2.3.0. To specify the version for installation, the command can be changed to “conda install tensorflow-gpu==1.15” where version 1.15 is the version used in this project. It is recommended to try version

1.15 and above to check whether the version still can be used with TensorFlow Object Detection API.

```
conda create --name tf_gpu  
  
activate tf_gpu  
  
conda install tensorflow-gpu
```

Figure 4.8: List of Command to Create A Virtual Environment

4.1.5 Data Training Process

After setting up the virtual environment for the TensorFlow, the dataset can be trained in the environment. A python script named train.py is used to start the training process for the SSD Mobilenet V2 Quantized. As the script is running, the program will load for about two to three minutes to collect the related data before proceed the training process. The training process takes about 12 hours to train the model to ensure the loss in the training session consistently drop below to below the value of 2. The time taken for the process is based on the dataset prepared earlier as shown in Figure 4.9.

4.1.6 TensorFlow Raspberry Pi

TensorFlow file cannot be directly apply to raspberry pi as the raspberry pi has very low computing power. The other method that available is converting the TensorFlow file to TensorFlow Lite. TensorFlow Lite is very suitable for low computing and mobile devices that want to run image processing. In the process of converting TensorFlow to TensorFlow Lite, TensorFlow Lite Optimizing Converter (TOCO) is used to convert the frozen graph we just exported into a model that can be used by TensorFlow Lite.

```

Administrator: Anaconda Prompt (anaconda3) (2) - python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/ssd_mobilenet...
INFO:tensorflow:global step 15971: loss = 1.4829 (1.225 sec/step)
I0127 20:59:40.730974 13148 learning.py:512] global step 15971: loss = 1.4829 (1.225 sec/step)
INFO:tensorflow:global step 15972: loss = 1.8795 (1.331 sec/step)
I0127 20:59:42.065155 13148 learning.py:512] global step 15972: loss = 1.8795 (1.331 sec/step)
INFO:tensorflow:global step 15973: loss = 1.5617 (1.332 sec/step)
I0127 20:59:43.399915 13148 learning.py:512] global step 15973: loss = 1.5617 (1.332 sec/step)
INFO:tensorflow:global step 15974: loss = 2.9193 (1.279 sec/step)
I0127 20:59:44.682125 13148 learning.py:512] global step 15974: loss = 2.9193 (1.279 sec/step)
INFO:tensorflow:global step 15975: loss = 1.4786 (1.210 sec/step)
I0127 20:59:45.894217 13148 learning.py:512] global step 15975: loss = 1.4786 (1.210 sec/step)
INFO:tensorflow:global step 15976: loss = 1.7117 (1.204 sec/step)
I0127 20:59:47.100749 13148 learning.py:512] global step 15976: loss = 1.7117 (1.204 sec/step)
INFO:tensorflow:global step 15977: loss = 2.9287 (1.217 sec/step)
I0127 20:59:48.318521 13148 learning.py:512] global step 15977: loss = 2.9287 (1.217 sec/step)
INFO:tensorflow:global step 15978: loss = 1.5421 (1.247 sec/step)
I0127 20:59:49.568752 13148 learning.py:512] global step 15978: loss = 1.5421 (1.247 sec/step)
INFO:tensorflow:global step 15979: loss = 1.8540 (1.207 sec/step)
I0127 20:59:50.778544 13148 learning.py:512] global step 15979: loss = 1.8540 (1.207 sec/step)
INFO:tensorflow:global step 15980: loss = 2.5271 (1.201 sec/step)
I0127 20:59:51.981249 13148 learning.py:512] global step 15980: loss = 2.5271 (1.201 sec/step)
INFO:tensorflow:global step 15981: loss = 2.3526 (1.201 sec/step)
I0127 20:59:53.183092 13148 learning.py:512] global step 15981: loss = 2.3526 (1.201 sec/step)
INFO:tensorflow:global step 15982: loss = 1.9394 (1.300 sec/step)
I0127 20:59:54.485291 13148 learning.py:512] global step 15982: loss = 1.9394 (1.300 sec/step)
INFO:tensorflow:global step 15983: loss = 1.4821 (1.216 sec/step)
I0127 20:59:55.704046 13148 learning.py:512] global step 15983: loss = 1.4821 (1.216 sec/step)
INFO:tensorflow:global step 15984: loss = 1.8103 (1.218 sec/step)
I0127 20:59:56.923976 13148 learning.py:512] global step 15984: loss = 1.8103 (1.218 sec/step)
INFO:tensorflow:global step 15985: loss = 2.7004 (1.242 sec/step)
I0127 20:59:58.168109 13148 learning.py:512] global step 15985: loss = 2.7004 (1.242 sec/step)
INFO:tensorflow:global step 15986: loss = 1.8496 (1.206 sec/step)
I0127 20:59:59.377372 13148 learning.py:512] global step 15986: loss = 1.8496 (1.206 sec/step)
INFO:tensorflow:global step 15987: loss = 1.9655 (1.196 sec/step)
I0127 21:00:00.574450 13148 learning.py:512] global step 15987: loss = 1.9655 (1.196 sec/step)
INFO:tensorflow:global step 15988: loss = 2.1449 (1.202 sec/step)
I0127 21:00:01.778122 13148 learning.py:512] global step 15988: loss = 2.1449 (1.202 sec/step)
INFO:tensorflow:global step 15989: loss = 1.8234 (1.208 sec/step)
I0127 21:00:02.988797 13148 learning.py:512] global step 15989: loss = 1.8234 (1.208 sec/step)
INFO:tensorflow:global step 15990: loss = 1.7020 (1.208 sec/step)
I0127 21:00:04.200155 13148 learning.py:512] global step 15990: loss = 1.7020 (1.208 sec/step)
INFO:tensorflow:global step 15991: loss = 2.1339 (1.203 sec/step)
I0127 21:00:05.404701 13148 learning.py:512] global step 15991: loss = 2.1339 (1.203 sec/step)
INFO:tensorflow:global step 15992: loss = 1.4654 (1.212 sec/step)
I0127 21:00:06.619320 13148 learning.py:512] global step 15992: loss = 1.4654 (1.212 sec/step)
INFO:tensorflow:global step 15993: loss = 1.1423 (1.206 sec/step)
I0127 21:00:07.827089 13148 learning.py:512] global step 15993: loss = 1.1423 (1.206 sec/step)
INFO:tensorflow:global step 15994: loss = 1.6067 (1.206 sec/step)
I0127 21:00:09.036486 13148 learning.py:512] global step 15994: loss = 1.6067 (1.206 sec/step)
INFO:tensorflow:global step 15995: loss = 2.1370 (1.205 sec/step)
I0127 21:00:10.243746 13148 learning.py:512] global step 15995: loss = 2.1370 (1.205 sec/step)
INFO:tensorflow:global step 15996: loss = 1.8346 (1.210 sec/step)
I0127 21:00:11.456161 13148 learning.py:512] global step 15996: loss = 1.8346 (1.210 sec/step)
INFO:tensorflow:global step 15997: loss = 3.1648 (1.203 sec/step)
I0127 21:00:12.661574 13148 learning.py:512] global step 15997: loss = 3.1648 (1.203 sec/step)
INFO:tensorflow:global step 15998: loss = 1.7041 (1.208 sec/step)
I0127 21:00:13.872534 13148 learning.py:512] global step 15998: loss = 1.7041 (1.208 sec/step)
INFO:tensorflow:global step 15999: loss = 2.3127 (1.203 sec/step)
I0127 21:00:15.078181 13148 learning.py:512] global step 15999: loss = 2.3127 (1.203 sec/step)
INFO:tensorflow:global step 16000: loss = 1.9143 (1.207 sec/step)
I0127 21:00:16.285976 13148 learning.py:512] global step 16000: loss = 1.9143 (1.207 sec/step)

```

Figure 4.9: Training Process using Anaconda Prompt

4.2 Python software programming

Python is a popular programming language. It was created by Guido van Rossum (Rossum, 2001), and released in 1991. Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

The Python coding uses a TensorFlow Lite model to perform object detection and image classification on a live webcam. It draws boxes and scores around the objects of interest in each frame from the webcam. To improve FPS, the webcam object runs in a separate thread from the main program using OpenCV. The coding is divided into six main sections:

Section 1: to initiate python coding using TensorFlow Lite

```
# Import packages
import os
import argparse
import cv2
import numpy as np
import sys
import time
from threading import Thread
import importlib.util
from Ultrasonic import ultrasonic
from Motor import motor
import RPi.GPIO as GPIO
```

Section 2: to define VideoStream class to handle streaming of video from webcam in separate processing thread

```
ult = ultrasonic()
mot = motor()
x = input("Ascending(A) or Descending(D): ")
if ( x == 'A'):
    stair = 'Ascending'
elif (x == 'D'):
    stair = 'Descending'
class VideoStream:
    """Camera object that controls video streaming from the PiCamera"""
    def __init__(self,resolution=(640,480),framerate=30):
        # Initialize the PiCamera and the camera image stream
        self.stream = cv2.VideoCapture(0)
        ret = self.stream.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))
        ret = self.stream.set(3,resolution[0])
        ret = self.stream.set(4,resolution[1])

        # Read first frame from the stream
        (self.grabbed, self.frame) = self.stream.read()
        # Variable to control when the camera is stopped
        self.stopped = False
    def start(self):
        # Start the thread that reads frames from the video stream
```

```

    Thread(target=self.update,args=()).start()
    return self
def update(self):
    # Keep looping indefinitely until the thread is stopped
    while True:
        # If the camera is stopped, stop the thread
        if self.stopped:
            # Close camera resources
            self.stream.release()
            return

        # Otherwise, grab the next frame from the stream
        (self.grabbed, self.frame) = self.stream.read()
def read(self):
    # Return the most recent frame
    return self.frame
def stop(self):
    # Indicate that the camera and thread should be stopped
    self.stopped = True

```

Section 3: to define and parse input arguments in the TensorFlow Lite

```

parser = argparse.ArgumentParser()
parser.add_argument('--modeldir', help='Folder the .tflite file is located in',
                    required=True)
parser.add_argument('--graph', help='Name of the .tflite file, if different than detect.tflite',
                    default='detect.tflite')
parser.add_argument('--labels', help='Name of the labelmap file, if different than labelmap.txt',
                    default='labelmap.txt')
parser.add_argument('--threshold', help='Minimum confidence threshold for displaying detected objects',
                    default=0.5)
parser.add_argument('--resolution', help='Desired webcam resolution in WxH. If the webcam does not support the resolution entered, errors may occur.',
                    default='1280x720')
parser.add_argument('--edgetpu', help='Use Coral Edge TPU Accelerator to speed up detection',
                    action='store_true')
args = parser.parse_args()

MODEL_NAME = args.modeldir
GRAPH_NAME = args.graph
LABELMAP_NAME = args.labels
min_conf_threshold = float(args.threshold)

```

```

resW, resH = args.resolution.split('x')
imW, imH = int(resW), int(resH)
use_TPU = args.edgetpu

```

Section 4: to import TensorFlow libraries and VideoStream initialisation

If tfLite_runtime is installed, import interpreter from tfLite_runtime, else import from regular tensorflow

If using Coral Edge TPU, import the load_delegate library

```
pkg = importlib.util.find_spec('tfLite_runtime')
```

```
if pkg:
```

```
    from tfLite_runtime.interpreter import Interpreter
```

```
    if use_TPU:
```

```
        from tfLite_runtime.interpreter import load_delegate
```

```
else:
```

```
    from tensorflow.lite.python.interpreter import Interpreter
```

```
    if use_TPU:
```

```
        from tensorflow.lite.python.interpreter import load_delegate
```

If using Edge TPU, assign filename for Edge TPU model

```
if use_TPU:
```

```
    # If user has specified the name of the .tflite file, use that name, otherwise use default
```

```
'edgetpu.tflite'
```

```
    if (GRAPH_NAME == 'detect.tflite'):
```

```
        GRAPH_NAME = 'edgetpu.tflite'
```

Get path to current working directory

```
CWD_PATH = os.getcwd()
```

Path to .tflite file, which contains the model that is used for object detection

```
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,GRAPH_NAME)
```

Path to label map file

```
PATH_TO_LABELS = os.path.join(CWD_PATH,MODEL_NAME,LABELMAP_NAME)
```

Load the label map

```
with open(PATH_TO_LABELS, 'r') as f:
```

```
    labels = [line.strip() for line in f.readlines()]
```

Have to do a weird fix for label map if using the COCO "starter model" from

https://www.tensorflow.org/lite/models/object_detection/overview

First label is '???' which has to be removed.

```
if labels[0] == '???':
```

```
    del(labels[0])
```

Load the Tensorflow Lite model.

```

# If using Edge TPU, use special load_delegate argument
if use_TPU:
    interpreter = Interpreter(model_path=PATH_TO_CKPT,
                             experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
    print(PATH_TO_CKPT)
else:
    interpreter = Interpreter(model_path=PATH_TO_CKPT)
interpreter.allocate_tensors()

# Get model details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]
floating_model = (input_details[0]['dtype'] == np.float32)
input_mean = 127.5
input_std = 127.5

# Initialize frame rate calculation
frame_rate_calc = 1
freq = cv2.getTickFrequency()

# Initialize video stream
videostream = VideoStream(resolution=(imW,imH),framerate=30).start()
time.sleep(1)

#for frame1 in camera.capture_continuous(rawCapture,
format="bgr",use_video_port=True):
while True:
    # Press 'q' to quit
    if cv2.waitKey(1) == ord('q'):
        break
    x_centre = 0
    img_centre = 0
    object_present = 0
    sleep = 1.5

    forward = ult.F_distance()
    bottom = ult.B_distance()
    right = ult.R_distance()
    left = ult.L_distance()

    # Start timer (for calculating frame rate)
    t1 = cv2.getTickCount()

```

```
# Grab frame from video stream
frame1 = videostream.read()

# Acquire frame and resize to expected shape [1xHxWx3]
frame = frame1.copy()
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
frame_resized = cv2.resize(frame_rgb, (width, height))
input_data = np.expand_dims(frame_resized, axis=0)
```

Section 5: Normalisation pixel values and object detection

```
# Normalize pixel values if using a floating model (i.e. if model is non-quantized)
if floating_model:
    input_data = (np.float32(input_data) - input_mean) / input_std

# Perform the actual detection by running the model with the image as input
interpreter.set_tensor(input_details[0]['index'],input_data)
interpreter.invoke()

# Retrieve detection results
boxes = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding box
coordinates of detected objects
classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class index of detected
objects
scores = interpreter.get_tensor(output_details[2]['index'])[0] # Confidence of detected
objects
#num = interpreter.get_tensor(output_details[3]['index'])[0] # Total number of detected
objects (inaccurate and not needed)

# Loop over all detections and draw detection box if confidence is above minimum
threshold
for i in range(len(scores)):
    if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0) and labels[int(classes[i])]
    == stair):

        # Get bounding box coordinates and draw box
        # Interpreter can return coordinates that are outside of image dimensions, need to
        force them to be within image using max() and min()
        ymin = int(max(1,(boxes[i][0] * imH)))
        xmin = int(max(1,(boxes[i][1] * imW)))
        ymax = int(min(imH,(boxes[i][2] * imH)))
        xmax = int(min(imW,(boxes[i][3] * imW)))
        cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)
        object_present = 1
```

```

    # Draw label
    object_name = labels[int(classes[i])] # Look up object name from "labels" array
using class index
    label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'
    labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX,
0.7, 2) # Get font size
    label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close
to top of window
    cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0],
label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) # Draw white box to put label text
in
    cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX,
0.7, (0, 0, 0), 2) # Draw label text

#calculate centre rectangle
x_centre = (xmin + xmax)/2
x_centre = int(x_centre)
cv2.line(image, (x_centre, 0), (x_centre, imH), (255, 0, 0), 2)

#calculate centre image
img_centre = imW/2
img_centre = int(img_centre)
cv2.line(image, (img_centre-20, 0), (img_centre-20, imH), (0, 0, 255), 2)
cv2.line(image, (img_centre+20, 0), (img_centre+20, imH), (0, 0, 255), 2)

####Robot Movement#####
if (object_present == 1):
    if ((forward > 6.0 and left > 6.0 and right > 6.0 and bottom < 11.0 )and(x_centre
> img_centre-20 and x_centre < img_centre+20)):
        mot.Forward()

    elif (x_centre < img_centre-20):
        left = ult.L_distance()
        if (left < 6.0):
            mot.Forward()
        else:
            mot.Left()
    elif (x_centre > img_centre+20):
        right = ult.L_distance()
        if (right < 6.0):
            mot.Forward()
        else:
            mot.Right()

```

```

elif ((x_centre > img_centre-20) and (x_centre < img_centre+20)):
    mot.Forward()
    forward = ult.F_distance()
    bottom = ult.B_distance()
    if (forward < 6.0):
        print('Ascending')
        break
    elif (bottom > 11.0 and x == 'D'):
        print('Descending')
        break
if (object_present == 0):
    if (bottom > 11.0 and x == 'D' ):
        print('Descending')
        break
    elif (right < 6.0):
        mot.Left()
    elif (left < 6.0):
        mot.Right()
    else :
        mot.Forward()
elif((forward < 6.0 or bottom > 11.0) and (object_present == 1)):
    break

```

Section 6: to draw framerate in corner of frame

```

cv2.putText(frame,'FPS:
{0:.2f}'.format(frame_rate_calc),(30,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,0),
2,cv2.LINE_AA)

```

```

# All the results have been drawn on the frame, so it's time to display it.
cv2.imshow('Object detector', frame)

```

```

# Calculate framerate
t2 = cv2.getTickCount()
time1 = (t2-t1)/freq
frame_rate_calc= 1/time1

```

```

# Clean up
cv2.destroyAllWindows()
videostream.stop()
GPIO.cleanup()

```

4.3 Experiment Setup

Before proceed to obtain the result in this project, a few preparations have to be made in order to get the correct data. The first preparation is to model the casing of the Rovision robot to hold the components in the right place. Solidwork software is used to design a simple casing that can placed the component and then the model is print out using 3d printer. It took 13 hours to finish printing the model. The circuit in Figure 3.2 is applied into the casing so that the camera and the ultrasonic sensors placed in the right position.

```
#calculate centre rectangle
x_centre = (xmin + xmax)/2
x_centre = int(x_centre)
cv2.line(image, (x_centre, 0), (x_centre, imH), (255, 0, 0), 2)

#calculate centre image
img_centre = imW/2
img_centre = int(img_centre)
cv2.line(image, (img_centre-3, 0), (img_centre-3, imH), (0, 0, 255), 2)
cv2.line(image, (img_centre+3, 0), (img_centre+3, imH), (0, 0, 255), 2)
```

Figure 4.10: Coding for Adding Line to the Bounding Box and Frame

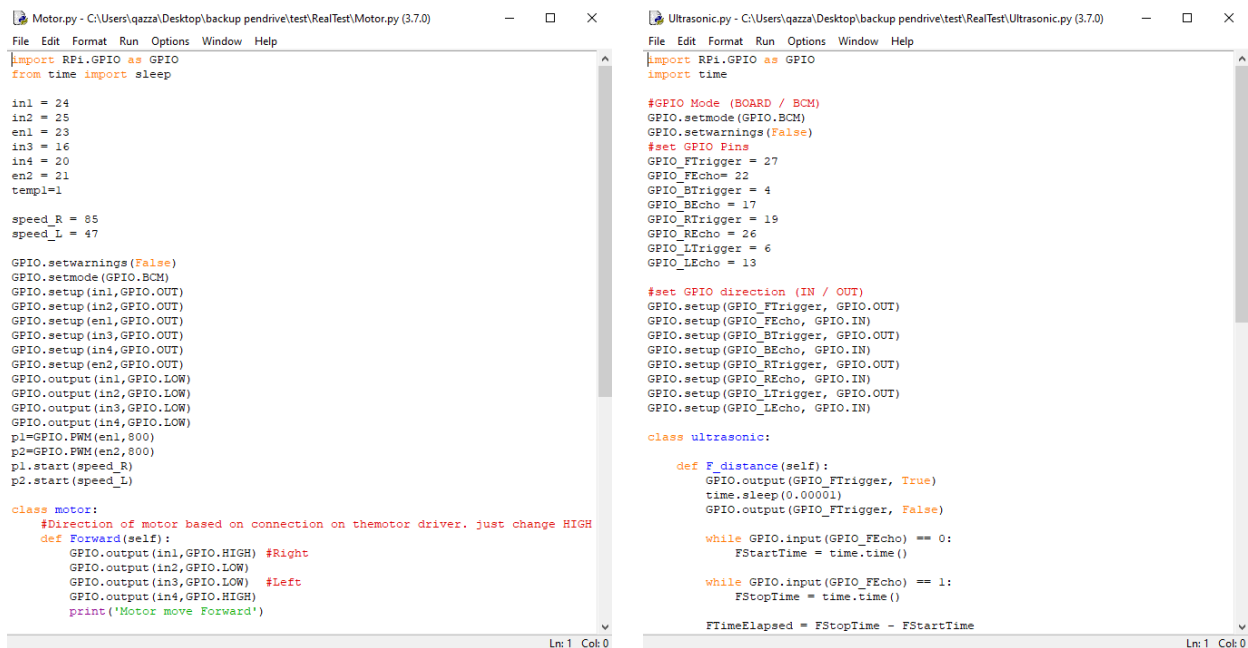


Figure 4.11: Motor.py and Ultrasonic.py Python Script

In order to create a robot that can move toward the desired destination, the python script need to be edited to make the robot know the position. After inserting the object detection API in the raspberry pi, a python file named “TFLite_detection_webcam.py” has been changed by adding code that can create a center line for the bounding box and the frame. The red lines indicate the center line for the frame while the blue line is the center line for the bounding box as shown in Figure 4.10.

In addition, extra script is created for the ultrasonic and motor to work with “TFLite_detection_webcam.py” script. The file has been named “Ultrasonic.py” and “Motor.py” as shown in Figure 4.11. The next step is to find the suitable place to test the system accuracy. A random place can be used to test the system whether staircase is present or not. In this project, the experiment is done at staircase at Block D Mahallah Ali as we want to test either the Rovision robot can detect and move toward the stair or it fail to detect. Figure 4.12 show some of the stairs where the Rovision is test out.

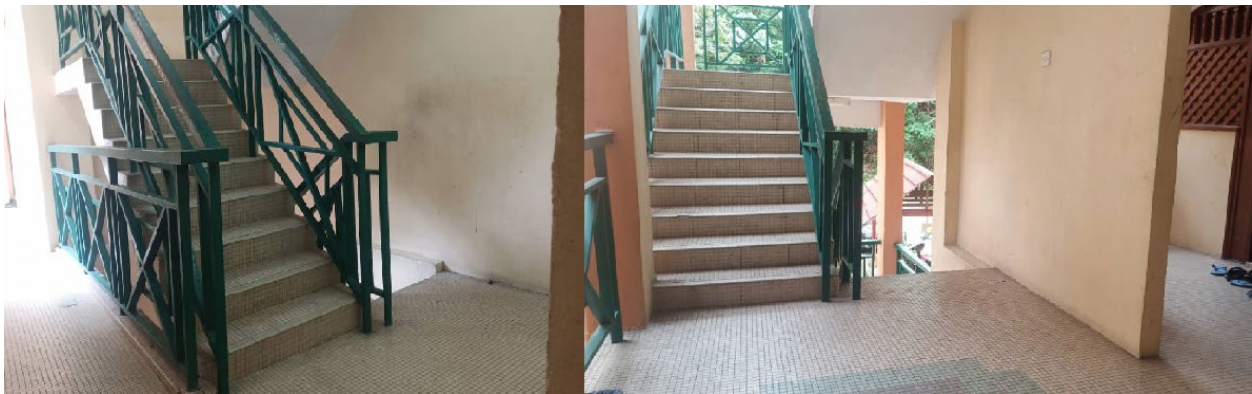


Figure 4.12: Experiment Venue at Mahallah Ali, IIUM

Chapter 5: Results and Analysis

5.1 Overview

In this chapter, results from the Rovision experimentation are discussed. The performance of the trained model of object detection has been observed based on the input and output given from the training data. A system can be observed through the efficiency of the machine to detect targeted object precisely. The result can be seen through the webcam output displayed on the computer screen while the robot is running. The trained data also obtain through the TensorBoard website prepared by TensorFlow while training the data. By setting up the number of epoch and layers of the neural network, it will affect the performance of the neural network model either to be more accurate or less accurate.

5.2 Results and Analysis

5.2.1 Accuracy of Ultrasonic Sensors

The ultrasonic sensor is one of the most important components in executing this project. Having accurate sensors is crucial as it affect the reliability of the machine. As ultrasonic sensor is used to determine the distance between the machine and the obstacle, it needs to calculate the distance precisely to avoid machine to hit the obstacle.

Four different distances have been setup to test the accuracy of the ultrasonic sensors. The distance used in this test will be used in the Rovision. Table 5.1 illustrate the result obtained from the accuracy test.

The accuracy of all ultrasonic sensors obtained from the test illustrate in Table 5.1 shows that the ultrasonic sensors are in good condition and can detect the correct distance accurately. 90% accuracy can indicate that the sensors can perform well for this project.

Table 5.1: The Result for Ultrasonic Sensor Accuracy Test

Ultrasonic sensor	Distance (5.0 cm)	Distance (10.0 cm)	Distance (20.0 cm)	Distance (40.0 cm)	Accuracy (%)
Front	4.5 cm	9.3 cm	19.0 cm	39.7 cm	94.25
Bottom	4.9 cm	10.0 cm	20.2 cm	40.0 cm	99.25
Right	4.8 cm	10.2 cm	19.8 cm	39.8 cm	98.00
Left	4.8 cm	10.0 cm	20.1 cm	39.0 cm	98.50

5.2.2 TensorFlow Training Result

In training the ssd mobilenet object detection model for ascending and descending staircase detection, the neural network for the deep learning has been setup in the script file name “ssd mobilenet v2 quantized 300x300 coco” which represent the name of the model. In the script, the layer for the neural network has been set to 6 layers referred to Figure 5.1 with batch size of 6 and 20k steps. The epoch for this model can be identify by dividing the number of steps with total image in train dataset which is 800 and multiply with batch size. The epoch is 125.

```

anchor_generator {
  ssd_anchor_generator {
    num_layers: 6
    min_scale: 0.2
  }
}

```

Figure 5.1: Layer can be determine by this line of code

The ssd mobilenet object detection model was trained using TensorFlow software where the best result of the model took about 12 hours of training to ensure the total loss in the training to be below two. It is the most suitable value for the loss as it indicates the model is perfectly trained. If the loss is too lower than two, the model might be overtrained and if it higher than two, it might be not enough trained and has problem in accuracy to detect staircase. The model then has been tested the accuracy by importing random stair image. Table 5.2 and Figure 5.2 illustrate the total loss value drop as the train session start. The

value of the total loss is taken every 1000 steps in the training session. A step is one operation to update the weights of the model. The number of steps is exactly the number of times the weights will be updated by the optimizer. It is evident from Figure 5.3 that the error minimization is reduced towards zero. The result of the test model can be seen in Figure 5.4, and Figure 5.5.

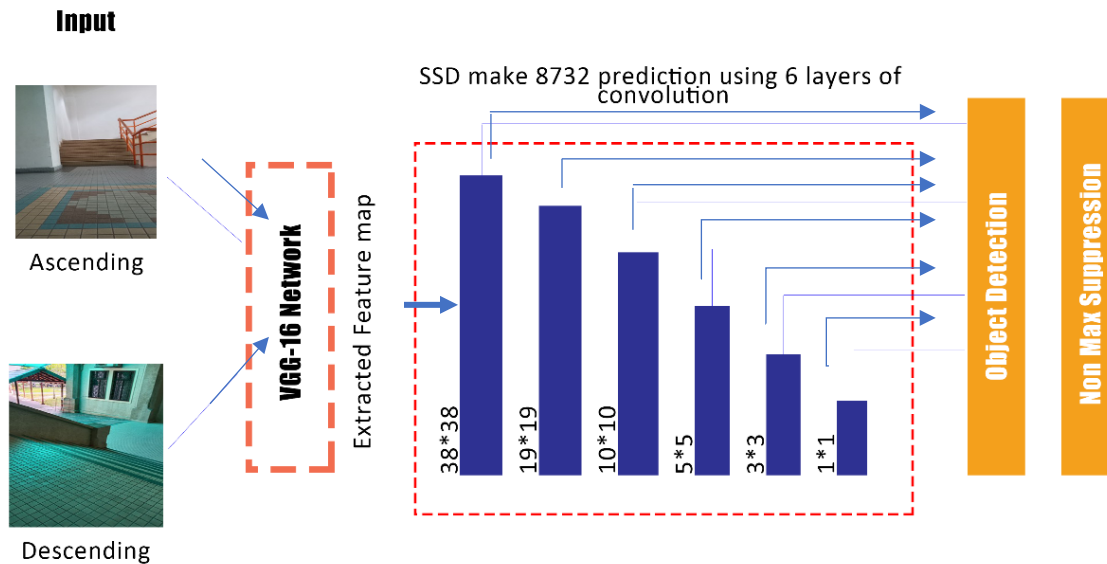


Figure 5.2: SSD Structure with 6 Convolution Layers for Revision project



Figure 5.3: Graph of Error Minimisation

Table 5.2: Total Loss Value Every 1000 Steps

Step	Total Loss
0	18.56
1000	6.004
2000	5.013
3000	4.388
4000	4.019
5000	3.845
6000	3.72
7000	3.543
8000	3.444
9000	3.373
10000	3.272
11000	3.163
12000	3.101
13000	3.049
14000	3.015
15000	2.956
16000	2.888
17000	2.695
18000	2.236
19000	1.988
20000	1.988

The ssd mobilenet object detection model was trained until 20,000 steps with the total loss = 1.988. At 20,000 steps, it has the lower total loss which mean the system has high accuracy in detecting the staircase compared to previous steps. Continuing the training might lead to overtrained model that can cause error in detecting staircase.

The results of the training indicate that the model has been well trained in the TensorFlow software as the accuracy in detecting the desired object mostly around 90% and above as it can be referred to Figure 5.4 that depict 96% and 100% accuracy for descending staircase detection and Figure 5.5 that depict 94% and 100% accuracy for ascending staircase detection. Thus, the model can be used to implement in the Rovision robot as the accuracy of the model can make Rovision more accurate and reliable to the visually impaired people.

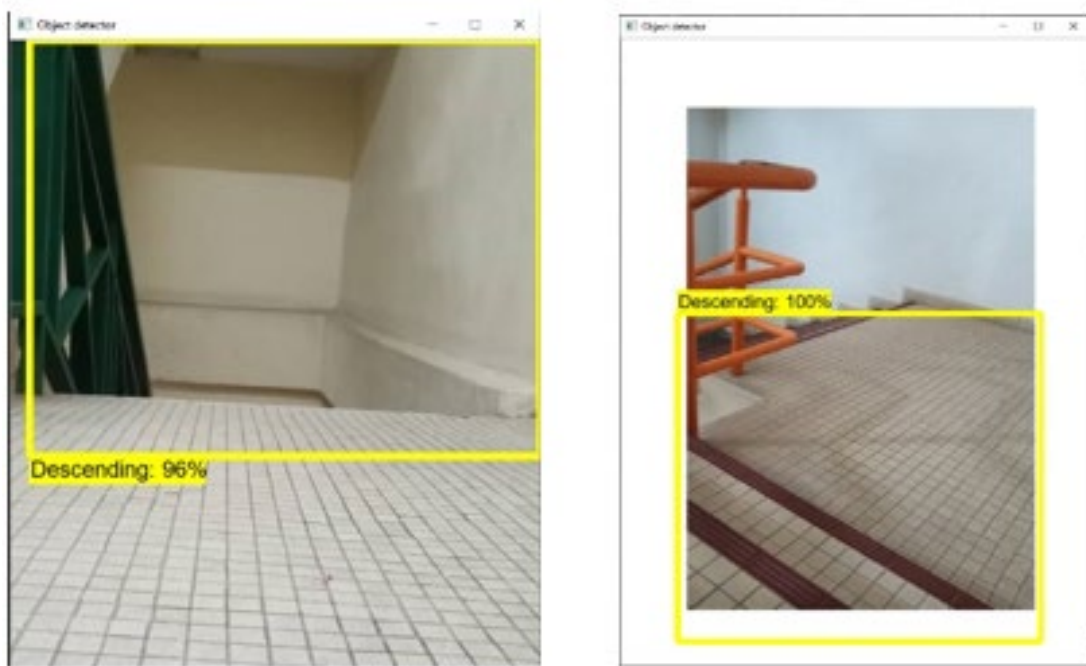


Figure 5.4: Result of Detecting Descending Staircase After Training the Model

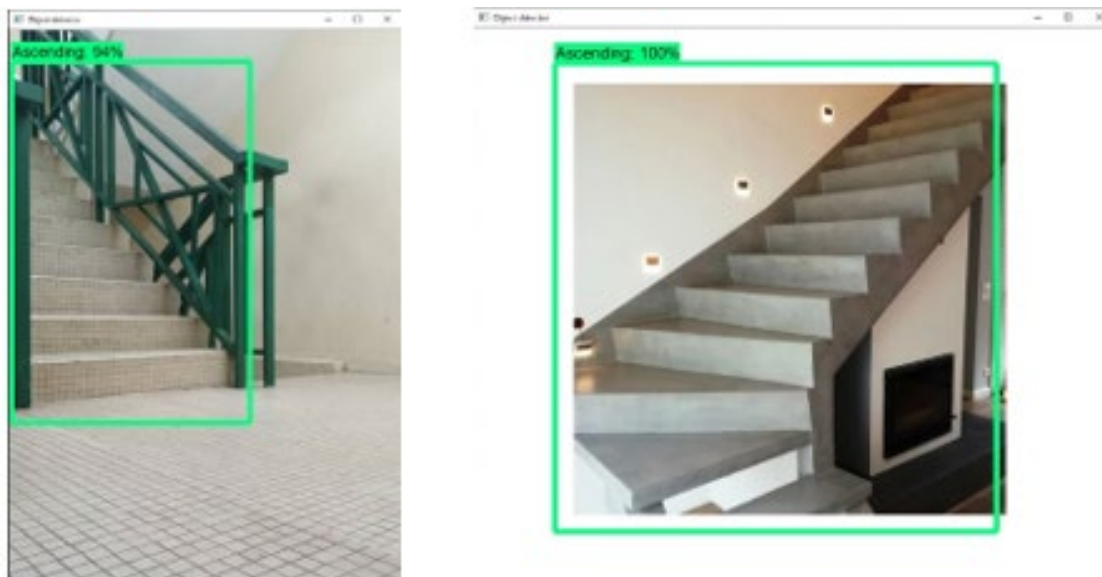


Figure 5.5: Result of Detecting Ascending Staircase After Training the Model

5.2.3 Performance of TensorFlow Lite on Raspberry

Pi (Camera)

The used of TensorFlow Lite is to make the object detection algorithm can be used in small computing power devices. After obtaining the result of training model, the frozen graph of the trained model was converted to TensorFlow Lite files where it will be used in the raspberry pi.

Ten videos have been recorded for every five times trial of Rovision detecting ascending staircase and 5 times trial for detecting descending staircase using Rovision webcam. Each length of the video is set to 1 minute and the frame is taken for every second in the video which is 60 frames. Every frame taken from the video will be analyze the detection of the system toward the staircase either it can detect correctly or there is still error in detecting staircase.

The accuracy of the system is calculated using the formula (7) which use the value from true positive, true negative, false positive and false negative that can be referred in Chapter 3, Section 3.6. Table 5.3 and Table 5.4 illustrates the result of the system accuracy based on ascending and descending conditions.

Table 5.3: The Accuracy of Ascending Staircase

Ascending	True Positive	True Negative	False Positive	False Negative	Accuracy (%)
Test 1	60	0	0	0	100.00
Test 2	54	0	0	6	90.00
Test 3	60	0	0	0	100.00
Test 4	60	0	0	0	100.00
Test 5	56	0	0	4	93.33
Average Percentage					96.66

Table 5.4: The Accuracy of Descending Staircase

Descending	True Positive	True Negative	False Positive	False Negative	Accuracy (%)
Test 1	56	0	0	4	93.33
Test 2	53	2	0	5	91.66
Test 3	55	0	0	5	91.66
Test 4	60	0	0	0	100.00
Test 5	56	3	0	1	98.33
Average Percentage					94.96

The result in the Table 5.3 and Table 5.4 shows that the Rovision has successfully detect the ascending and descending staircase with high average percentage of accuracy which are 96.66% for ascending detection and 94.96% for descending detection. Thus, the results indicate that Rovision can accurately distinguish between ascending and descending staircase with less error occur during the process.

5.2.4 Performance of TensorFlow Lite on Raspberry

Pi (Movement)

Referring to the System Flowchart in Figure 3.2, The main purpose of this project is to make the Rovision robot to be able to detect and move toward the staircase whenever the staircase present. The system also needs to be able to distinguish between the ascending and descending staircase. As for the result, the robot only performs the detection and movement part only. The robot will break the loop whenever it in front of the stair (ascending or descending). The movement of the robot either it moves forward, turn right, or left and move backward was recorded in the terminal in raspberry pi. As can be seen in Figure 5.6 and Figure 5.7, Rovision will navigate by determine the distance between the blue line and the red line shown in both figures. The blue line represents the

center line of the Area of Detection Box (green box) and red line represent the center of the frame. As the blue line goes to the left of red line, the robot will turn left and if blue line on the right side, robot will act vice versa. The decision of Rovision will be encountered by the ultrasonic sensors signal whenever it detects obstacle. For instance, if the blue line is on left side of red line, but ultrasonic sensor detects present of obstacle like wall, the Rovision will keep moving forward until it save to turn.

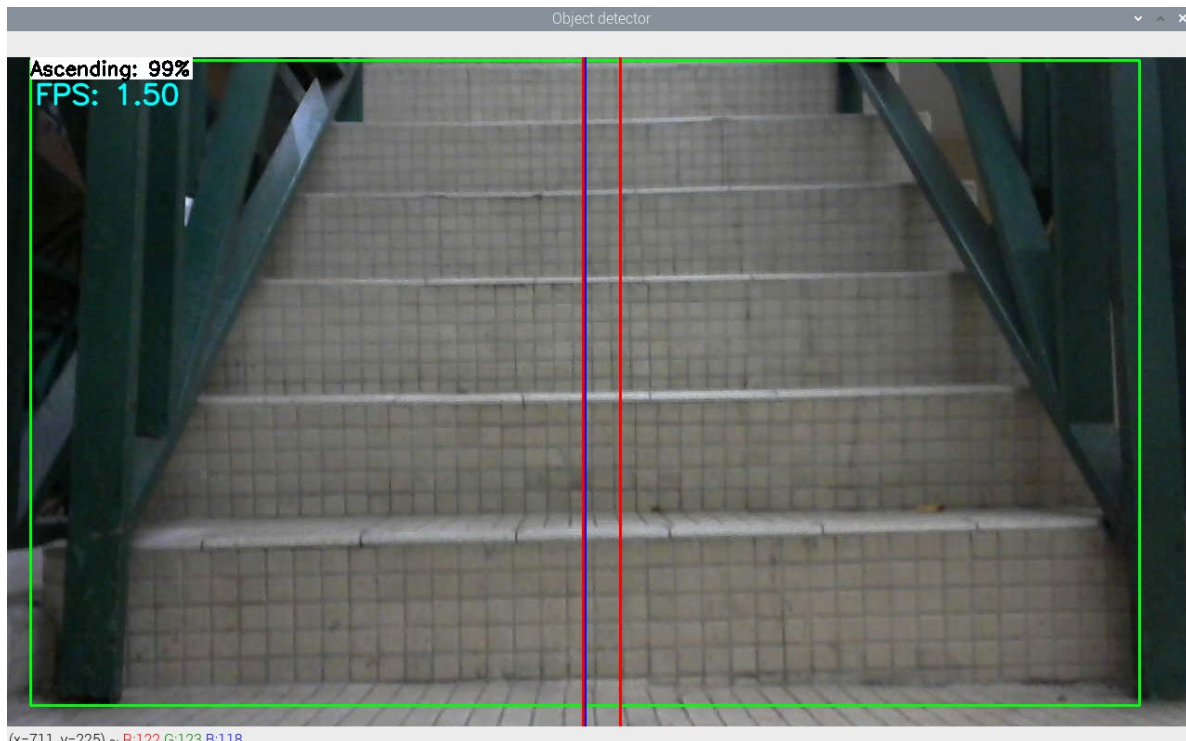


Figure 5.6: Result of Detecting Ascending Staircase

Two red line indicates the tolerance given to make Rovision run smoothly. The tolerance helps the Rovision to move forward whenever the blue line is between both red line, which mean the Rovision will not only move left and right all time. The tolerance given is 20 pixels from center of the frame. Between two red line, it consists total of 40 pixels. The tolerance can be modified by changing the value in the coding that can be referred in Section 4.2: Python Programming as shown in Figure 5.7.

```

Motor move Left
Corrupt JPEG data: 3 extraneous bytes before marker 0xd6
Corrupt JPEG data: 4 extraneous bytes before marker 0xd0
Corrupt JPEG data: 2 extraneous bytes before marker 0xd6
Corrupt JPEG data: 1 extraneous bytes before marker 0xd2
Corrupt JPEG data: 4 extraneous bytes before marker 0xd0
Corrupt JPEG data: 2 extraneous bytes before marker 0xd7
Corrupt JPEG data: 2 extraneous bytes before marker 0xd0
Corrupt JPEG data: 2 extraneous bytes before marker 0xd1
Corrupt JPEG data: 2 extraneous bytes before marker 0xd6
Corrupt JPEG data: 1 extraneous bytes before marker 0xd7
Corrupt JPEG data: 1 extraneous bytes before marker 0xd6
Corrupt JPEG data: 1 extraneous bytes before marker 0xd1
Motor move Left
Corrupt JPEG data: 2 extraneous bytes before marker 0xd4
Corrupt JPEG data: 2 extraneous bytes before marker 0xd5
Corrupt JPEG data: 1 extraneous bytes before marker 0xd4
Corrupt JPEG data: 2 extraneous bytes before marker 0xd2
Corrupt JPEG data: 2 extraneous bytes before marker 0xd0
Corrupt JPEG data: 3 extraneous bytes before marker 0xd1
Corrupt JPEG data: 1 extraneous bytes before marker 0xd0
Corrupt JPEG data: 1 extraneous bytes before marker 0xd6
Corrupt JPEG data: 2 extraneous bytes before marker 0xd5
Corrupt JPEG data: 1 extraneous bytes before marker 0xd4
Corrupt JPEG data: 4 extraneous bytes before marker 0xd5
Motor move Forward
Ascending
(tflite1-env) pi@pi:~/tfliteTest $

```

Figure 5.7: Result as Revision Closed to Ascending Staircase

The terminal will print out the result whenever it found the ascending or descending staircase. The maximum FPS (frame per second) that raspberry pi 4 B+ can give is only 4FPS which is lower to obtain smooth real-time video. It is recommended to have at least 10 FPS to perform smooth image processing to get better result especially for mobile robot as shown in Figure 5.7.

Nevertheless, the lack of FPS not affect the performance of the Revision as the Revision be able to accurately detect the ascending and descending staircase with high accuracy and be able to reach the staircase. Figure 5.8 and Figure 5.9 depicts the on-going process of detecting and movement of the robot toward the ascending and descending staircase respectively.



Figure 5.8: Result of Detecting Descending Staircase

```

Motor move Left
turn left
Corrupt JPEG data: 1 extraneous bytes before marker 0xd6
Corrupt JPEG data: 1 extraneous bytes before marker 0xd6
Corrupt JPEG data: 2 extraneous bytes before marker 0xd3
Corrupt JPEG data: 1 extraneous bytes before marker 0xd5
Corrupt JPEG data: 2 extraneous bytes before marker 0xd3
Corrupt JPEG data: 1 extraneous bytes before marker 0xd2
Corrupt JPEG data: 3 extraneous bytes before marker 0xd2
Corrupt JPEG data: 1 extraneous bytes before marker 0xd1
Corrupt JPEG data: 1 extraneous bytes before marker 0xd1
Corrupt JPEG data: 1 extraneous bytes before marker 0xd1
Corrupt JPEG data: 1 extraneous bytes before marker 0xd2
Motor move Left
turn left
Corrupt JPEG data: 1 extraneous bytes before marker 0xd3
Corrupt JPEG data: 1 extraneous bytes before marker 0xd7
Corrupt JPEG data: 1 extraneous bytes before marker 0xd0
Corrupt JPEG data: 1 extraneous bytes before marker 0xd0
Corrupt JPEG data: 1 extraneous bytes before marker 0xd2
Corrupt JPEG data: 1 extraneous bytes before marker 0xd6
Corrupt JPEG data: 2 extraneous bytes before marker 0xd5
Corrupt JPEG data: 1 extraneous bytes before marker 0xd1
Corrupt JPEG data: 2 extraneous bytes before marker 0xd5
Corrupt JPEG data: 2 extraneous bytes before marker 0xd4
Corrupt JPEG data: 2 extraneous bytes before marker 0xd6
Descending
(tfllite1-env) pi@pi:~/tfliteTest $

```

Figure 5.9: Result as Revision Detect Descending Staircase

5.3 Discussion

Based the result form the experiment, it shows that the robot can perform all the predetermined tasks successfully. The Rovision robot able to detect and differentiate between the ascending and descending staircase accurately using the object detection image processing model. With all the components work in excellent condition, the objective of this project can be achieved. From the ultrasonic sensor's accuracy result in Table 5.1, all the four ultrasonic sensors have the accuracy of 90% and above which indicates that it can give accurate responses to the Rovision whenever any obstacle or staircase present in front of the robot. For the result in Table 5.2, the losses produce by the system drastically drop at the first 1000 steps which is from 18.56 to 6.004 and gradually decrease to below two points at the 20000th steps. The reduction in the loss shows that the system become more stable and be able to detect the staircase accurately. The losses produce in the training result is the additional from localization loss and the classification loss. Localization loss is where the error that system produce to locate the position of the staircase in the dataset and classification loss is the error of system to correctly classify the class of the object detected by the system. The decreasing of the losses indicates the system become more accurate to detect stair.

As the raspberry pi performance different from the normal computer, another accuracy test has been done to check the accuracy of the machine to detect the staircase, ascending and descending. The result in Table 5.3 and Table 5.4 of staircase average percentage of accuracy on raspberry pi shows Rovision can detect ascending staircase with 96.66% accuracy and descending staircase with 94.96% accuracy. It means Rovision can handle simple task like detecting the ascending and descending staircase precisely. Therefore, with the implementation of image processing in Rovision really aids the robot to improve detection of obstacle and staircase. It shows that to make the robot more accurate in detection object, it is recommended to use different type of sensors so that the machine can rely on all the data from different sensors. For example, this project use camera as main sensor and ultrasonic sensor as second sensor. As the camera cannot be rotated, it only covers the front part of the robot to "see". With the help of ultrasonic sensors, Rovision can make the right decision whenever it detect staircase either move forward, turn, reverse, or stop. Even though the FPS in the real time test can only reach up to 4FPS, it only affects the movement of the Rovision in reaching the destination, but Rovision still has a very good performance in detecting the staircase.

Chapter 6: Conclusion & Future Work

6.1 Conclusion

In conclusion, the objectives of this project have been achieved. The Rovision robot now has the capability to identify and differentiate between ascending and descending staircase and can maneuver itself toward the targeted staircase. The result indicates that the Rovision robot has potential in helping the visually impaired people to navigate and finding the staircase in future. Based on the result of training and result on the raspberry pi, there are plenty of room for improvement that can be done to make the robot robust. The webcam's FPS can be one of the components that can be improved. To smooth out the movement of the robot, the system required at least 10 to 15 FPS. A device called Coral USB accelerator can be used to increase the FPS on the raspberry pi to 24 to 30 FPS but the disadvantage is the device is very expensive.

6.2 Future Work

Future work is the place where the idea for improvement is delivered. In my opinion, a PID controller can be included in future as the robot can be much more stable in navigating to staircase. The PID controller can be used whenever the blue line is between the red line. A motor decoder and gyro sensor also can be included in the machine to be able for the user to control the speed of the robot movement and to enhance the stability during navigating. Thus, by doing such improvement mention above, Rovision is ready to be used by the visually impaired people to aids them on their daily activities.

REFERENCE

- Adiwahono, A. H., Saputra, B., Chang, T. W., & Yong, Z. X. (2014). Autonomous stair identification, climbing, and descending for tracked robots. *2014 13th International Conference on Control Automation Robotics and Vision, ICARCV 2014, 2014*(December), 48-53. <https://doi.org/10.1109/ICARCV.2014.7064278>
- Alake, R. (2020, February 6). *A Beginner's Introduction To TensorFlow Lite / by Richmond Alake / Towards Data Science*. Towards Data Science. <https://towardsdatascience.com/a-beginners-introduction-to-tensorflow-lite-924320deed5>
- Ertuğrul, Ö. F. (2018). A novel type of activation function in artificial neural networks: Trained activation function. *Neural Networks, 99*, 148-157. <https://doi.org/10.1016/j.neunet.2018.01.007>
- Farooq, U., Amar, M., Ul Haq, E., Asad, M. U., & Atiq, H. M. (2010). Microcontroller based neural network controlled low cost autonomous vehicle. *ICMLC 2010 - The 2nd International Conference on Machine Learning and Computing*, 96-100. <https://doi.org/10.1109/ICMLC.2010.71>
- Feng, J., & Lu, S. (2019). Performance Analysis of Various Activation Functions in Artificial Neural Networks. *Journal of Physics: Conference Series, 1237*(2). <https://doi.org/10.1088/1742-6596/1237/2/022030>
- Forson, E. (2017, November 18). *Understanding SSD MultiBox – Real-Time Object Detection In Deep Learning / by Eddie Forson / Towards Data Science*. Towards Data Science. <https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab>
- Galvez, R. L., Bandala, A. A., Dadios, E. P., Vicerra, R. R. P., & Maningo, J. M. Z. (2019). Object Detection Using Convolutional Neural Networks. *IEEE Region 10 Annual International Conference, Proceedings/TENCON, 2018-Octob*(October), 2023-2027. <https://doi.org/10.1109/TENCON.2018.8650517>
- Łabęcki, P., Walas, K., & Kasinski, A. (2011). Autonomous stair climbing with multisensor feedback. *IFAC Proceedings Volumes (IFAC-PapersOnline), 44*(1 PART 1), 8159-8164. <https://doi.org/10.3182/20110828-6-IT-1002.02747>
- Mahmood, N. H., Ahmad, A. H., & Omar, C. (2015). *Sonar Assistive Device for Visually Impaired People Jurnal Teknologi Sonar Assistive Device for Visually Impaired People. April*. <https://doi.org/10.11113/jt.v73.4404>
- Masood, M. U., Sami, M. A., Sohail, H., Mujtaba, M., Siddique, M. A., Akram, H., Rashid, N., Tiwana, M. I., & Iqbal, J. (2017). Design and development of a semi-autonomous stair climbing robotic platform for rough terrains. *International Conference on Control, Automation and Systems, 2017-Octob*(Iccas), 212-217. <https://doi.org/10.23919/ICCAS.2017.8204443>

- Medina-Santiago, A., Camas-Anzueto, J. L., Vazquez-Feijoo, J. A., Hernández-De León, H. R., & Mota-Grajales, R. (2014). Neural control system in obstacle avoidance in mobile robots using ultrasonic sensors. *Journal of Applied Research and Technology*, 12(1), 104-110. [https://doi.org/10.1016/S1665-6423\(14\)71610-4](https://doi.org/10.1016/S1665-6423(14)71610-4)
- Mihankhah, E., Kalantari, A., Aboosaeedan, E., & Taghirad, H. D. (2009). *Autonomous Staircase Detection and Stair Climbing for a Tracked Mobile Robot using Fuzzy Controller*. 1980-1985.
- Nada, A., Mashelly, S., Fakhr, M. A., & Seddik, A. F. (2015). *Effective Fast Response Smart Stick for Blind People*. April. <https://doi.org/10.15224/978-1-63248-043-9-29>
- Nagasawa, K., Fukumura, N., & Uno, Y. (2005). *A Forward-Propagation Learning Rule for Acquiring Inverse Models in Multilayered Neural Networks*. 88(2), 1066-1074. <https://doi.org/10.1002/ecjb.20148>
- Roumeliotis, S. I., Michael, C., & Matthies, L. (2002). *Multi-Sensor , High Speed Autonomous Stair Climbing*. 733-742.
- Rossum, G. V. (2001), An Introduction to Python, Release 2.2.2, page 1-111, Network Theory Ltd.
- Sumit, S. (2018, December 16). *A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way | by Sumit Saha | Towards Data Science*. Towards Data Science. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- Toha, S. F and Tokhi, M. O (2008), MLP and Elman recurrent neural network modelling for TRMS, In Proceedings of 7th IEEE International Conference on Cybernetic Intelligent Systems (CIS08), London, UK, 9-10 September 2008, pp. 391-396.
- Thu, A. M. M., Aung, M. T. S., & Okada, T. (2019). Autonomous Stairs Ascending and Descending Algorithm for Tri-Star Wheeled Robot. *ICARM 2018 - 2018 3rd International Conference on Advanced Robotics and Mechatronics*, 328-333. <https://doi.org/10.1109/ICARM.2018.8610676>
- Tydén, A., & Olsson, S. (2020). *Edge Machine Learning for Animal Detection, Classification, and Tracking*. 59.
- Wolff, C. (2017, December 5). *Comparing Image-Classification Systems: Custom Vision Service vs. Inception | CSE Developer Blog*. <https://devblogs.microsoft.com/cse/2017/12/05/comparing-transfer-learning-systems-custom-vision-service-vs-inception-vs-mobilenet/>

ABOUT THE AUTHORS



Siti Fauziah Toha (B.Eng'03-M'06-PhD'10), is currently an Associate Professor at the Department of Mechatronics Engineering, International Islamic University Malaysia (IIUM). She received B. Eng (Hons) in Electrical and Electronics Engineering from University Technology Petronas and later received MSc from Universiti Sains Malaysia in electrical engineering. She was then completed her Ph.D in Automatic Control and Systems Engineering from The University of Sheffield in

2010.

She was later join the Perusahaan Otomobil Nasional Berhad (PROTON) Malaysia as a control expert consultant, working on AI-based battery management system for electric vehicle. Her current research interest spanning over the Modelling and Analysis of Complex System (MACS), Control Algorithms and Artificial Intelligence Optimisation, Assistive Devices and Bio-inspired robotics, Green Renewable Energy. Dr Toha is a senior member of IEEE and also a Professional Engineer with the Board of Engineers Malaysia (BEM) as well as a Chartered Engineer with Engineering Council, The Institution of Engineering and Technology, United Kingdom. She also served as the Engineering Accreditation Council (EAC) panel for BEM. She is also an active member of Young Scientist Network, Academy of Sciences Malaysia (YSN-ASM) and appointed as Co-chair for Science Policy Working Group (2018).



Ahmad Syahrin Idris (B.Eng'03-M.Phil'11-Dr.Eng'18), is currently an Assistant Professor at the School of Electronics and Computer Science, University of Southampton Malaysia (UoSM). He received his B. Eng (Hons) in Electrical and Electronics Engineering from University Technology Petronas, Malaysia and received his M.Phil from The University of Sheffield, UK in Electronic and Electrical Engineering. He later received his PhD in Opto-electronics from Kyushu University, Japan.

After his B.Eng degree, he joined Intel as a Product Development Engineer specializing in developing design-for-test solutions for Intel chipset products. While in the UK, he was also a researcher at the University of Sheffield specializing in the fabrication and characterization of III-V semiconductors for APD and PIN photodetectors. After his M.Phil degree, he joined Freescale Semiconductor as a Senior Test Development Engineer developing test solutions for automotive and industrial microcontrollers. His current research interests are in fabrication and characterization of opto-electronic devices and developing design-for-test solutions for microelectronic circuits. Dr. Idris is also a

Professional Engineer with the Board of Engineers Malaysia and The Institute of Engineers Malaysia.



Abdur Razzaq Abd Halim was born on December 14th, 1997 at General hospital, Kuala Lumpur, Malaysia. He received his Bachelor of Engineering in Mechatronic (B. Eng) in 2021 and currently pursued Master of Sciences (Msc) in the field of Electronics Engineering, at International Islamic University Malaysia.

His research interests related to development of fluidic channels and packaging for screening of COVID-19. His recent work is about designing the packaging of the sensors while developing the readout circuit on detecting the COVID-19 virus. This research able to assist in preventing the spread of the infectious viruses as the area of interest in this research is related to early stage patient of COVID-19.

AI-BASED LEVEL DETECTION AND OPTIMISATION OF ASSISTIVE ROBOT MANEUVERABILITY

Assistive devices for blind and visually impaired people are one of the technologies that famous among the researchers. There are many devices has been developed by the researcher in order to aid the visually impaired people to move around. One of the challenges faces by the blind and visually impaired people is stair. Using traditional method, the present of the staircase cannot be detected in a safe distance. Rovision is one of the devices that has the capability to guide the user to move to desired place without hitting any obstacle or object. In mobile robot, sensors play important roles in guiding the robot by sending the data of the surrounding for the robot to execute the action. Camera and ultrasonic sensor are two sensors that use in the Rovision to navigate the robot in safe distance while detecting the staircase. Image processing is one of the best methods in detecting staircase. It has a capability to learn by its own using the training dataset. By training the dataset, the system be able to identify the staircase and the position inside the camera frame to help Rovision to maneuver and guide the user safely. Rovision also use ultrasonic sensors to avoid obstacle in surrounding in order to have clear and safe path for the visually impaired person. This AI-Based Level Detection and Optimisation of Assistive Robot Maneuverability book will be useful for postgraduate students as well as final year undergraduate students researching on robotics area especially using the latest python software with focus on artificial intelligence techniques.

Centre for Professional Development (CPD)
International Islamic University Malaysia
Jalan Gombak,
Selangor Darul Ehsan,
MALAYSIA
Tel: +603-6421 5914/ Fax: +6421 5915
Email: admin_cpd@iium.edu.my
Website: www.iium.edu.my/centre/cpd

e ISBN 978-967-19026-5-3

