



**A DESIGN OF AUTONOMOUS REMOTE CONTROL CAR
USING CONVOLUTIONAL NEURAL NETWORK AND
HAAR-LIKE FEATURES CLASSIFIER**

**A final project report
presented to
the Faculty of Engineering**

By

**Andre Muslim
002201600006**

**in partial fulfillment
of the requirements of the degree
Bachelor of Science in Electrical Engineering**

**President University
January 2020**



**A DESIGN OF AUTONOMOUS REMOTE CONTROL CAR
USING CONVOLUTIONAL NEURAL NETWORK AND
HAAR-LIKE FEATURES CLASSIFIER**

**A final project report
presented to
the Faculty of Engineering**

By

**Andre Muslim
002201600006**

**in partial fulfillment
of the requirements of the degree
Bachelor of Science in Electrical Engineering**

**President University
January 2020**

DECLARATION OF ORIGINALITY

I declare that this final project report, entitled “A Design of Autonomous Remote Control Car Using Convolutional Neural Network and Haar-like Features Classifier” is my own original piece of work and, to the best of my knowledge and belief, has not been submitted, either in whole or in part, to another university to obtain a degree. All sources that are quoted or referred to are truly declared.

Cikarang, Indonesia, January 2020

A handwritten signature in black ink, consisting of a large, stylized 'M' with a vertical line extending downwards from its center, and a horizontal line at the top that curves to the right and ends in an arrowhead.

Andre Muslim

**A DESIGN OF AUTONOMOUS REMOTE CONTROL CAR
USING CONVOLUTIONAL NEURAL NETWORK AND
HAAR-LIKE FEATURES CLASSIFIER**

By

Andre Muslim

002201600006

Approved by



Iksan Bukhori, S.T., M.Phil
Final Project Supervisor



Antonius Suhartomo, Ph.D.
Head of Study Program
Electrical Engineering

ACKNOWLEDGMENTS

Alhamdulillahirabbil'alamin, praise to Allah SWT for all His guidance and smoothness so that I can finish this final project on time.

This thesis is dedicated to all people who always give me a lot of support and motivation, especially for my beloved parents Mr. Muslim and Mrs. Eti Wardanis who always concern about every path that I took also my brother Dicky who always allow me to use all his stuffs.

I would also thank my final project advisor, Mr. Iksan Bukhori S.T., M.Phil. for the advice, guidance, patience, and support to do the best for this final project. I wish to thank all Electrical Engineering lectures for their guidance since I begin my university life in President University until now.

Furthermore, I am feeling grateful to have all members of Electrical Engineering 2016 as my classmate, Deo Lumoindong, Jagat Kalam N., Indah Tria W., M. Arief Aryadi, M. Arwin Renardi, M. Yeza Baihaqi, and Wilbert Wijaya, for giving me endless support from the first day until end of my journey in this university. They had fulfilled my university life with wonderful and joyful story that I will always remember.

Last but not least, I would like to thank my beloved friends M. Arief Aryadi, who is my friend since senior high school that always supports my study and my school life, Jagat Kalam N., who always support my university life, M. Arwin R., Indah ria W., M. Yeza B., that have given me not only phisycal supports but also emotional supports in order to finish this final project.

Cikarang, January 2020,

Andre Muslim

APPROVAL FOR SCIENTIFIC PUBLICATION

I hereby, for the purpose of development of science and technology, certify and approve to give President University a non-exclusive royalty-free right upon my final project report with the title:

A DESIGN OF AUTONOMOUS REMOTE CONTROL CAR USING CONVOLUTIONAL NEURAL NETWORK AND HAAR-LIKE FEATURES CLASSIFIER

Along with the related software or hardware prototype (if needed). With this non-exclusive royalty-free right, President University is entitled to conserve, to convert, to manage in a database, to maintain, and to publish my final project report. These are to be done with the obligation from President University to mention my name as the copyright owner of my final project report.

Cikarang, 24 January 2020

A handwritten signature in black ink, consisting of a large circle with the letters 'm' and 'b' inside, followed by a vertical line.

Andre Muslim

002201600006

ABSTRACT

Autonomous vehicle is a type of vehicle that can drive safely without any human intervention. This safety is related with the capability of the vehicle to keep driving on its own track without disturbing the other lines, detect objects in front of it and estimate the distance to that object to prevent accident. However, there are only few researchers developing autonomous vehicle that can follow the predetermined path, detect objects, and estimate distance to said objects. In this final project, the author wants to make an autonomous remote control car that has those three features. This project develops autonomous remote control car that is controlled by convolutional neural network to keep the car in track. The device has features to classify three object classes (i.e. pedestrian, car, and stop sign) by using Haar-like classifier. In addition, the device can estimate the distance to the object by using pinhole imaging theory. The device takes images from a mobile phone attached to the car as its only input and processes the images in MATLAB2019a. This final project describes the entire process of its creation from hardware requirements, through the design of the control system, up to the selection and training of a convolutional neural network. The final device is able to follow the track with the accuracy ranging from 86.67% to 100.00% and classify three object classes with the accuracy ranging from 53.33% to 86.67%. In addition, the device can estimate the object distance with average error equals to 2.43 cm.

Keyword: Convolutional Neural Network, Haar-like Features Classifier, Object Distance Measurement, MATLAB 2019a.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION	1
1.1. Final Project Background	1
1.2. Problem Statement	2
1.3. Final Project Objectives	3
1.4. Final Project Scope and Limitation.....	3
1.5. Final Project Outline	4
CHAPTER 2 LITERATURE REVIEW AND DESIGN SPECIFICATION	5
2.1. Literature Review.....	5
2.1.1. Autonomous RC car	5
2.1.2. CNN.....	5
2.1.3. Haar-like features cascade classifier.....	10
2.1.4. Pinhole imaging theory.....	12
2.2. Software	13
2.2.1. MATLAB 2019a.....	13
2.2.2. Arduino IDE	15
2.2.3. IP Webcam android app	16
2.3. Hardware.....	17
2.3.1. RC car.....	18
2.3.2. Mobile phone.....	18
2.3.3. Arduino Nano	19
2.3.4. Slide DIP switch	20
2.3.5. Laptop.....	21
CHAPTER 3 DESIGN IMPLEMENTATION.....	22

3.1. Introductory Remarks	22
3.2. System Design	22
3.3. Hardware Implementation	24
3.3.1. Car	24
3.3.2. Controller and Arduino.....	24
3.4. Image Classification Algorithm	26
3.4.1. Image extraction	27
3.4.2. Path classification	28
3.4.3. Object classification and object distance measurement	30
3.5. Training Setup and Result.....	32
3.5.1. Path classification	32
3.5.2. Object classification	36
3.5.3. Object distance measurement	38
3.6. Arduino Program	39
CHAPTER 4 RESULT AND DISCUSSIONS	42
4.1. Results and Discussions	42
4.1.1. Path classification on the training path.....	42
4.1.2. Path classification on the custom path.....	43
4.1.3. Object classification	44
4.1.4. Object distance measurement	45
4.2. Device's Frailty Discussions.....	46
4.2.1. Reflected light.....	46
4.2.2. False positive	48
CHAPTER 5 CONCLUSIONS AND FUTURE DEVELOPMENTS	49
5.1. Conclusions.....	49
5.2. Future Developments	49

LIST OF FIGURES

Figure 1.1 SAE automation levels	2
Figure 2.1 Parts of ANN.....	6
Figure 2.2 The basic of CNN architecture [7]	7
Figure 2.3 Parts of CNN.....	7
Figure 2.4 Feature extraction process.....	8
Figure 2.5 Result of max pooling layer	9
Figure 2.6 Features in object classification algorithm [16].....	11
Figure 2.7 Source image and integral image	11
Figure 2.8 Degenerated decision trees [17].....	12
Figure 2.9 Pinhole imaging principle [21].....	12
Figure 2.10 MATLAB 2019a UI.....	14
Figure 2.11 Available apps on MATLAB 2019a	14
Figure 2.12 Image labeler app UI.....	15
Figure 2.13 Camera calibrator UI.....	15
Figure 2.14 Arduino IDE UI.....	16
Figure 2.15 IP webcam UI.....	17
Figure 2.16 IP webcam video preferences.....	17
Figure 2.17 RC car.....	18
Figure 2.18 Schematic of unmodified controller push button.....	18
Figure 2.19 Samsung J1 mini	19
Figure 2.20 Arduino nano board.....	19
Figure 2.21 Six pins slide DIP switch	20
Figure 2.22 Six pins DIP switch schematic	21
Figure 3.1 Overall system block diagram.....	23
Figure 3.2 Flow chart of autonomous RC car	23
Figure 3.3 Side looking device	24
Figure 3.4 Modified controller button	25
Figure 3.5 Controller as seen from	26
Figure 3.6 Image classification process.....	27
Figure 3.7 Video preferences on IP webcam app.....	27
Figure 3.8 Example of IP webcam's IP address.....	28
Figure 3.9 Code to grab an image frame from IP webcam.....	28

Figure 3.10 Code to specify main folder of training data.....	29
Figure 3.11 Code to initialize CNN.....	29
Figure 3.12 CNN design.....	30
Figure 3.13 Code to train Haar-like features cascade classifier	31
Figure 3.14 Code to estimate object distance	32
Figure 3.15 Training path.....	33
Figure 3.16 Car training results	38
Figure 3.17 Results of camera calibration.....	39
Figure 3.18 Arduino code.....	40
Figure 4.1 Example of custom path on carpet.....	43
Figure 4.2 Example of path in human eye perspective.....	47
Figure 4.3 Example of path in network's perspective	47
Figure 4.4 Example of false positive	48

LIST OF TABLES

Table 2.1 Example of Activation Function [13].....	10
Table 2.2 The Specification of Arduino Nano	20
Table 2.3 Acer Aspire E5-471G Specifications	21
Table 3.1 Arduino Pin Configurations	25
Table 3.2 IC Pin Reading	25
Table 3.3 Example of Training Image and Testing Image	34
Table 3.4 CNN Specifications	35
Table 3.5 Network Training Results.....	36
Table 3.6 Example of Objects Training Data	36
Table 3.7 Training Object Setup.....	37
Table 3.8 Haar-like Classifier Training Results	38
Table 3.9 MATLAB Image Class Interpretation.....	40
Table 4.1 Device Test Results on Trained Path.....	42
Table 4.2 Custom Path Testing Results.....	44
Table 4.3 Object Classification Test Results.....	44
Table 4.4 Object Distance Measurement Results	45

CHAPTER 1

INTRODUCTION

1.1. Final Project Background

In this century, machine involves in many aspects of human life. From time to time human always want to make machine more efficient and minimize human bias. So, human starts to build a machine that can work and learn without any human intervention. The only model that can learn which human has is its own brain. Artificial Intelligence (AI) is often used to describe machines (or computers) that mimic cognitive functions that humans associate with the human mind, such as learning and problem solving [1]. Many philosophers and scientist argue that with the current technology, it is reasonable to say AI is feasible within this century [2]. AI is basically everywhere around human's life and it can solve many problems. One of them is related with road safety.

According to the data from World Health Organization (WHO) China is the biggest country in term of total registered vehicles in 2016. About 294 million vehicles registered and WHO estimated that road traffic fatalities would be about 256,180 cases in year 2016 [3]. In addition, 1.35 million people around the world died each year because of car crashes. Moreover, car crashes is number 8th leading cause of death for people of all ages. One of the reason vehicle crashes is bad behavioral drivers which involved driving above the speed limit, and drink-driving [3]. So, human play a big role at the reason of a crash. Therefore, engineer around the world started to develop autonomous vehicle that uses AI technology.

Basically many features of vehicle have been automated, such as closing and opening doors, starting the engine, but the essence of driving task has not been changed that much [4]. The Society of Automotive Engineers (SAE) International classify autonomous vehicle into six different levels as shown in Figure 1. Currently, there are three companies that lead the autonomous car industry which are Tesla, Waymo, and Yandex. Until year 2019, the highest level of autonomous car they can produce is level four.

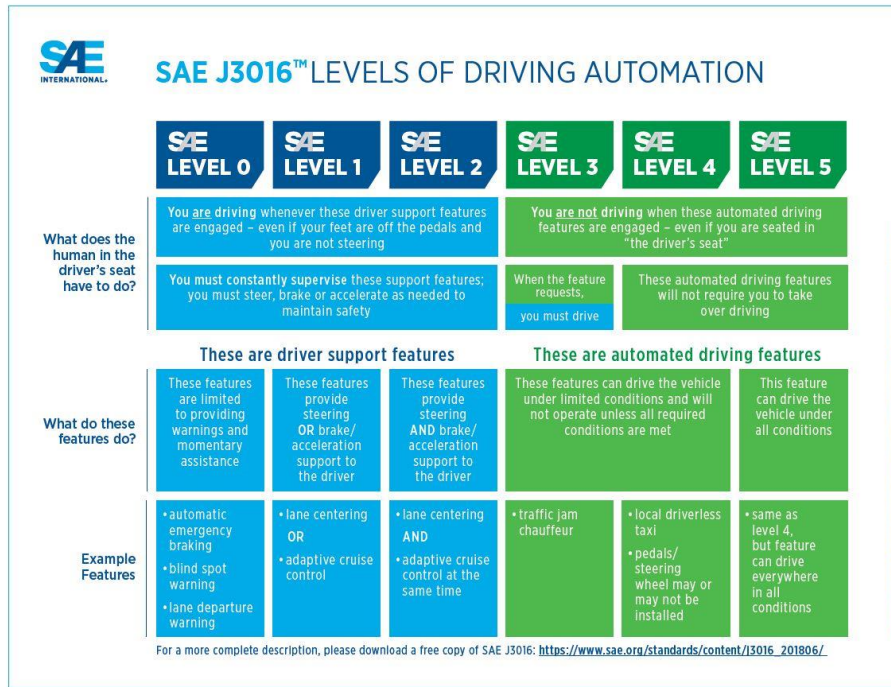


Figure 1.1 SAE automation levels

For example, the 2018 Tesla auto-pilot technology is considered as level three and it can operate anywhere as long as the condition is met. There are cars that already in level four of driving automation and is operating at only specific location, one of them is Waymo car. In this final project, the author wants to build the prototype of autonomous vehicle by using Remote Control (RC) car to imitate real vehicle with the level of driving automation is somewhere between level two and three. Since, the prototype cannot monitor all of its surroundings but it can recognize not only a car object therefore it fall in between level two and three. This final project aims to create an autonomous RC car that can operate under certain limited condition which will be covered later in this paper.

1.2. Problem Statement

One of the reason vehicle crashes is bad behavioral drivers which involved driving above the speed limit, and drink-driving [3]. Therefore, many engineers around the world start doing research about autonomous vehicle. However, there are only few researchers developing autonomous vehicle that can follow the predetermined path, detect objects, and estimate distance to said objects. Therefore, in this final project, the author wants to make an autonomous RC car that can drive through between two side lines and detect multiple

specific objects. In addition to the objects detection, the device also can estimate the distance to that objects.

1.3. Final Project Objectives

The final project objective is to design an autonomous RC car which is able to:

1. Automatically keep the RC car on the desired path by classifying the path between straight, left, or right.
2. Detecting object at the same time while the autonomous RC car is moving.
3. To additionally estimate the distance between the device to the object with at the maximum 2.5 cm average of error.

1.4. Final Project Scope and Limitation

The final project conducts under the following scopes:

- This final project discusses about the image processing and electrical modification used in obtaining the full autonomous RC car.
- The program of the device is developed in MATLAB 2019a and Arduino Integrated Development Environment (IDE).
- The Artificial Neural Network (ANN) used in the project is developed using the Deep Neural Network Toolbox.
- The Haar Cascade classifier is developed using the Image Processing Toolbox.
- The android app used in the project is IP Webcam app that can be downloaded on Google Playstore.
- The remote controller is controlled by using Arduino Nano which involve basic electrical modification

In conducting this research, there are several limitations to be considered:

- The car path should have two white colored side lines with distinct road color and it can be clearly seen by the mobile phone camera. Therefore, no glossy surface is allowed since it potentially read as white color by the mobile phone camera.
- The minimum width of the road is about 24.5 cm. This number is taken based on National Association of City Transportation Official urban street design guide that the ideal lane width is about 10 feet long [5]. Since the author uses the standard RC with

1:13 scale, therefore the ideal lane width is 23.5 cm. The author add one cm to give the RC car more space to navigate.

- The object that will be detected in this project is limited to specific object only. Furthermore, the object must be distinct in color compare to its environment.
- The distance of the RC car to the computer is no more than one meter. It is because in this project the remote control is the one that controls the car. Therefore, the distance limit is the same as standard 2.4 GHz remote control.

1.5. Final Project Outline

This final project report consists of five chapters and is outlined as follows:

1. Chapter 1 – Introduction

This chapter is an overview of the overall topic and the aim of the project. It consists of: Final Project Background; Final Project Objectives; Final Project Scope and Limitation; and Final Project Outline.

2. Chapter 2 – Literature Review and Design Specifications

This chapter present literature review from the existing researchers, and also describes about the software and its features, the design of autonomous RC car, and the design of ANN.

3. Chapter 3 – Design Implementations

This chapter covers the utilization techniques and the implementation of the software and hardware of this final project.

4. Chapter 4 – Result and Discussion

This chapter elaborates how the device works. It also presents data collected from the several trials, results of the project, some analysis, and discussions.

5. Chapter 5 – Conclusions and Recommendations

This chapter consists of conclusions obtained throughout the project implementation and recommendation for future development.

CHAPTER 2

LITERATURE REVIEW AND DESIGN SPECIFICATION

2.1. Literature Review

There are many kind of Machine Learning (ML) or ANN algorithms used to develop autonomous car, for example Support Vector Machine (SVM), probabilistic neural network, Convolutional Neural Network (CNN), etc. [6]. CNN is used in this project to develop the device because it is typically used to solve image-driven pattern recognition task. In addition, it is easy to get started with CNN because the architecture is simple yet precise [7]. This part explains the basic knowledge of all the methods including the CNN architecture involved to make the device.

2.1.1. Autonomous RC car

There are at least two researches did something quite similar to this final project. The first one is the project done by Chong Wen Yang [8]. This project used single board computer (i.e. Raspberry Pi 3) to process all of the input such as ultra-sonic sensor to detect obstacle, webcam for image processing, mobile phone to change from manual mode to auto mode, play station 4 controller to control the RC car when in manual mode, motor and servo for moving the RC car. It had feature to change lanes when there was an obstacle in front of the RC car and went back to original lane when it got through the obstacle. It also estimated the mid-point of its lane for keeping the RC car on the lane. However, since it was only using a single board computer to process all of the input, the RC car move really slow and often stopped working entirely. The second one is the project done by David Ungurean [9]. This is simpler than what Chong did because the inputs only consist of images from webcam. There was no object detection and lane changing algorithm. In addition, the network worked pretty well to predict its path and possible to work at 20 fps video feed with Raspberry Pi 3 as the main control board.

2.1.2. CNN

ANN is inspired by the biological neural network in human brain [10]. ANN has three layers which are input layer, hidden layer, and output layer. The basic ANN architecture is shown in Figure 2.1. The advantage of using ANN is that ANN has the ability to learn and

model real-life data. This is important because real-life data are messy and complicated. Thus, it is hard to define the relationship between its input and output. Moreover, ANN can generate useful information from complicated data and even can provide a prediction to what is likely to happen. Like many others machine learning algorithm, performance of ANN is also proportional to the amount of training data.

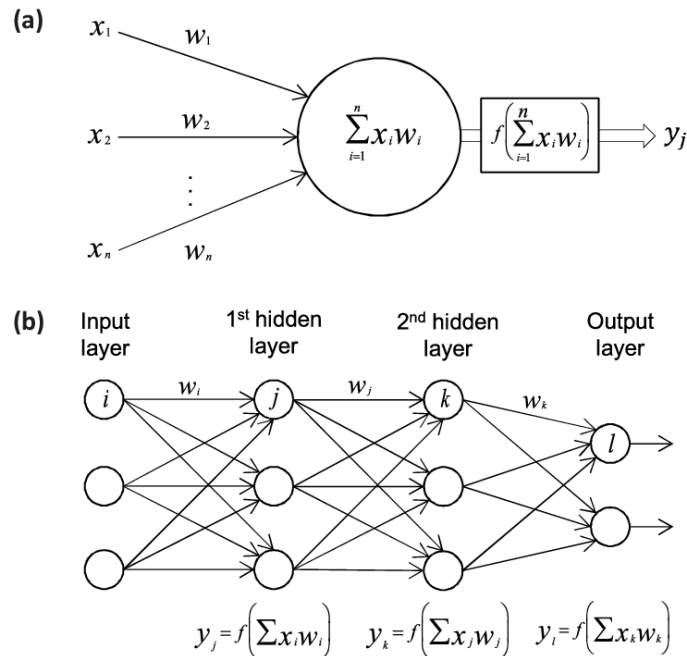


Figure 2.1 Parts of ANN: (a) artificial neuron or node; (b) feed forward fully connected multilayer [11]

ANN is constructed with many artificial neuron or node. Basically, neuron has two main parts which are summation part and activation function as shown in Figure 2.1 (a). Every neuron inputs are weighted with certain constant and this constant can be adjusted in the training process. Moreover, after each of the input is weighted, all of the inputs are summed before go to the activation function. Activation function is the one that determine the result of a node. Whether the result is real number or binary number (i.e. 0 or 1) depends on the activation function. There are many activation functions, for example sigmoid function, tanh function, Rectified Linear Unit (ReLU), etc. This activation function is typically determined by the network designer and it is free to choose depending on the situation. ANN is a set of interconnected node from input layer all the way through the output layer as can be seen in Figure 2.1 (b). CNN is one of the ANN architecture. There are several applications of CNN in the field, especially when it comes to image processing

like image classification, pattern recognition, etc. [12]. The basic CNN architecture is shown in Figure 2.2. The explanation of each layer is covered later in this chapter.

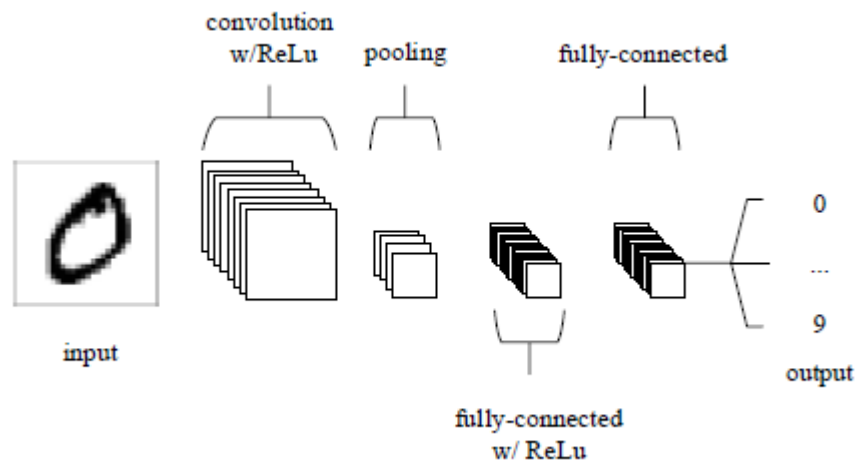


Figure 2.2 The basic of CNN architecture [7]

2.1.2.1. Convolution layer

The convolution layer is the one of the most important layer in CNN because it produces a feature map of an image input. Feature map is important because it collects the important information in the input image (i.e. horizontal line, vertical line) and neglect unimportant information (i.e. background noise) to save time and processing power. The term convolution refers to the mathematical combination of two functions to produce a third function. CNN use a so-called filter and then applied it to the input to produce a feature map. The representation of filter looks like in Figure 2.3.

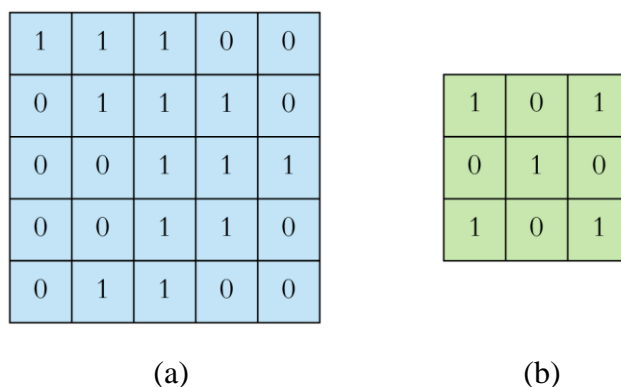


Figure 2.3 Parts of CNN: (a) input pixels; (b) filter or kernel

Note that the size of a filter is changeable. Basically, filter is used to extract feature from the input pixel by using element-wise matrix multiplication and the sum result is stored

into feature map as shown in Figure 2.4. Therefore, the values inside a filter are changeable for feature extraction purpose and the filter values are determined by training.

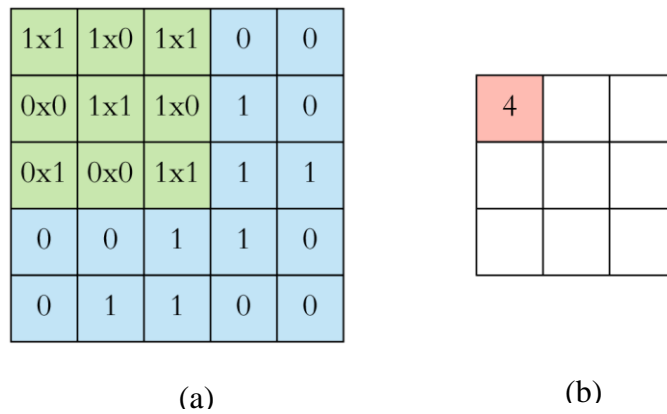


Figure 2.4 Feature extraction process: (a) input multiplied by filter; (b) resulting feature map

Actually the value of feature map is not taken directly from the element wise of the input and filter. Since the values of input are not always in binary, it uses normalization as part of the pre-processing step so that all the data points are rely on the same scale. For example, the input values are ranging from zero to a thousand, then normalization scales down the numerical data to be ranging from zero to one. This normalization is used to stabilize the numerical data meaning that there is no numerical data that might be very high compare to the others. The Then, the normalization output values go to ReLu first and then the result is stored in the feature map. However, Figure 2.4 omitted the ReLu function for simplification purpose. One convolutional layer might have many numbers of filters. All of the feature map from each of the filter would stack to each other. There are two parameters (i.e. stride and padding) needed to be chosen before applying the filter. Here is a simple explanation of those:

- Stride specifies how much pixel to move a filter at each step. The value of a stride determines the size of the feature map. The bigger the value of a stride the smaller the feature map would become because it skip over certain number of pixels.
- Padding is the additional pixels that surround the input. The value of padding is filled with zeros or the value on the edge. Padding is used to keep the dimension of the feature map the same size as the input.

2.1.2.2. Pooling layer

The purpose of pooling layer is to reduce the dimension of a feature map. Reducing the number of dimension of feature causes reduction of the parameters (i.e. weight) to save time and processing power. The common type of pooling method is max pooling which simply takes the maximum value in the pooling window. Figure 2.5 shows the example of max pooling layer in a feature map.

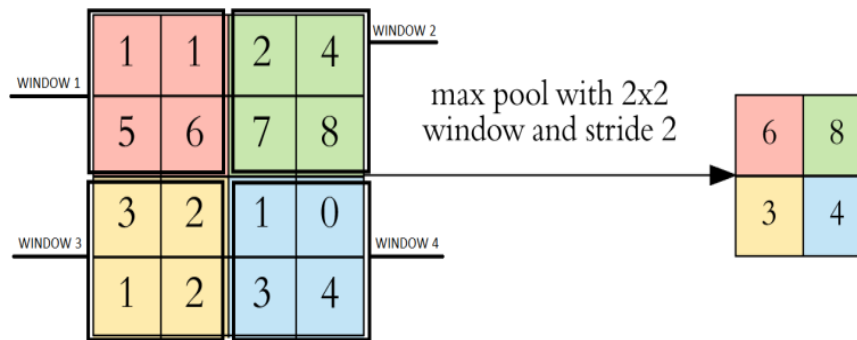


Figure 2.5 Result of max pooling layer

Each color denotes a different window. With 2x2 max pool size and stride two on the 4x4 feature map size the result is 2x2 sizes which actually halved each of the width and height. This is important to reduce number of parameters while keeping the important information on each of the feature map.

2.1.2.3. Fully connected layer

Fully connected layer is just a basic feed forward network. This layer combines all the features learned by the previous layers across the image to identify the larger patterns. It is possible to have more than one fully connected layer; however the last fully connected layer must be the same size as the number of classes in target data because it combines the features to classify the images. Usually, after fully connected layer there is a softmax layer which normalizes the output of the fully connected layer by using softmax activation function. The output of the softmax layer consists of positive numbers that sum to one, which can then be used as classification probabilities by the classification layer. All of the mentioned activation functions are listed in Table 2.1.

Table 2.1 Example of Activation Function [13]

Function Name	Mathematical Expression
Sigmoid Function	$f(x) = \left(\frac{1}{1 + \exp^{-x}}\right)$
Tanh Function	$f(x) = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)$
ReLu Function	$f(x) = \begin{cases} x_i & \text{for } x_i > 0 \\ 0 & \text{for } x_i \leq 0 \end{cases}$
Softmax Function	$f(x) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$

2.1.3. Haar-like features cascade classifier

In this project, Haar-like feature cascade classifier or simply called Haar-like feature algorithm is used to detect specific object. Haar-like feature is a ML object detection algorithm generally used to detect object in an image or video [14]. This algorithm has two main points to operate quite fast. First, image representation using so-called “*integral image*” and this image representation allow the features to be computed really fast. Then, the so-called “*cascade*” method is a method that makes one classifier for each of the selected feature and applies all of it to the input images. Haar-like feature algorithm works by measuring the difference in each feature region (i.e. the difference in black and white region) as shown in Figure 2.6. Note that feature in this case is just like a filter in CNN and because it is pre-determined, there is no need to create feature map like in CNN. The process of filtering images using Haar-like features takes a lot of time. Therefore, an integral image method is proposed by Paul Viola and Michael Jones [15].

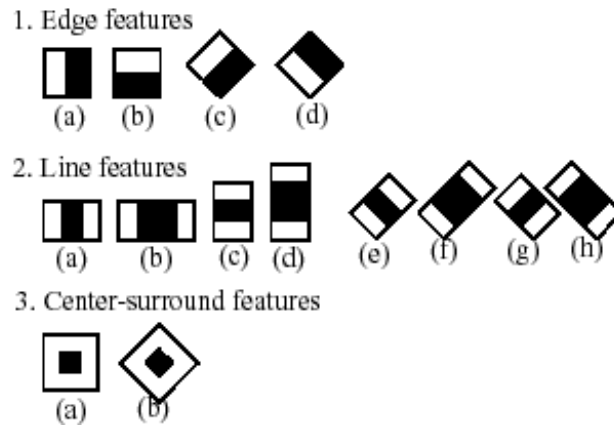


Figure 2.6 Features in object classification algorithm [16]

Integral images is the method to pre-compute an image and store it in an intermediate form that later on the calculation of feature can be done easily. Every new pixel in the intermediate form is the sum of the entire original pixel above and to the left. The representation of integral images is shown in Figure 2.7.

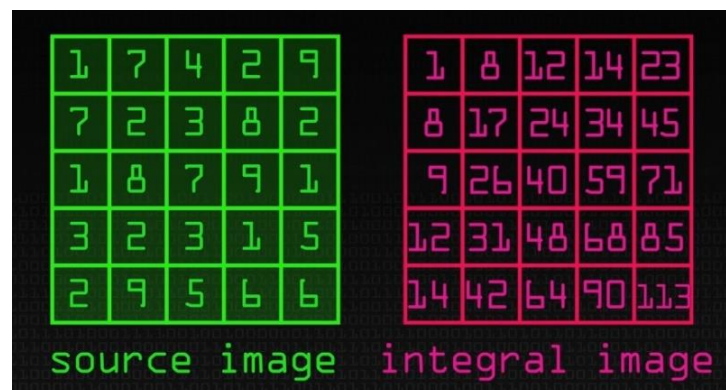


Figure 2.7 Source image and integral image

This integral image is really useful because it decrease the number of computation process. This is really important because the algorithm is not looking at the images one time but a thousand of times with many combinations of features in the training phase.

To train the algorithm, positive images (i.e. images contain the desired object) and negative images (i.e. images without the desired object) are needed [16]. In the training phase, the algorithm calculates all of the possible combination of feature and for a given data set it determines the best feature that separate the positives and negatives. After the algorithm found the best features that separate positives and negatives, then the algorithm used those features as a classifier to determine whether it contain the desired object or not by

evaluating each of the selected features. This checking process is using the so-called degenerate decision tree as shown in Figure 2.8 and the whole process named as cascade.

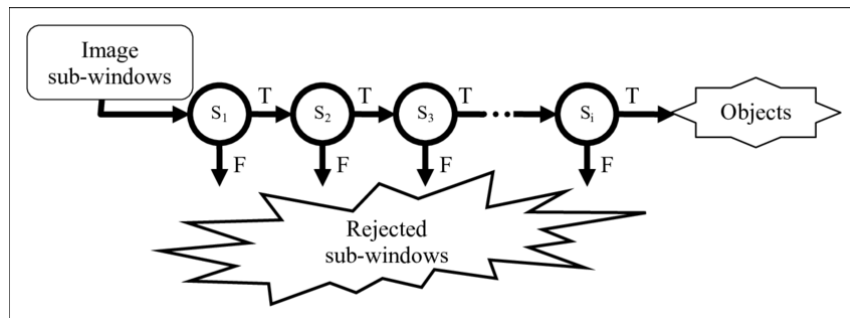


Figure 2.8 Degenerated decision trees [17]

So, the input image is being tested with selected feature and if it got pass then it goes to the next feature. If all of the features pass for certain image then it is likely that the image contain the desired object.

2.1.4. Pinhole imaging theory

There are several methods to estimate object distance from an image or images. For example, Joglekar [18] use several camera parameters such as height of camera, focal length, angle tilt of camera, pixel resolution to estimate object distance. However, the result is not good enough. This project only uses one camera on the car, therefore the method used is monocular vision and it based on the pinhole imaging theory [19]. Several researchers used this method, for example Peyman Alizadeh [20] and Yu-Tao Cao [21]. The maximum average of error of 2.5 cm as stated in Chapter I is inspired by the paper written by Peyman [20]. The pinhole imaging principle is shown in Figure 2.9.

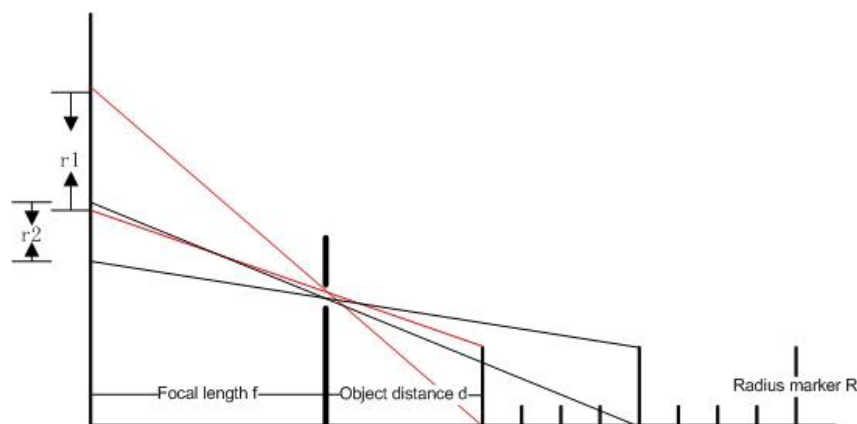


Figure 2.9 Pinhole imaging principle [21]

Note that the height of object in world coordinate system is R . On the other hand, r is the height of object in camera coordinate system. Moreover, focal length is denoted as f while distance is d . The author used width of the object as the base to calculate the object distance because it is easier to find the ratio between object width in image and object width in real-life rather than the ratio of object height. Therefore, the author replaced R with width of the object in world coordinate system which denoted by W and replaced r with width of object in camera coordinate system which denoted by w . Based on pinhole imaging principle and similar triangles, the relationship can be obtained as follows:

$$d = \frac{f}{\sqrt{\frac{w}{W}}} \quad (1)$$

Distance of object from the camera can be calculated using Eq. (1). However, the height or width of real object should be known for exact calculation [20]. Any error in each variable would imply to the distance result.

2.2. Software

In this final project there are three softwares used. The first one is MATLAB 2019a that used to process images from mobile phone in real-time. The second one is Arduino IDE to control the remote controller. The last one is IP Webcam that installed on the mobile phone to capture images.

2.2.1. MATLAB 2019a

The image processing algorithm is programmed using MATLAB 2019a. MATLAB has its own programming language called MATLAB. However, it is possible to change language in the setting menu. The User Interface (UI) of MATLAB 2019a is shown in Figure 2.10.

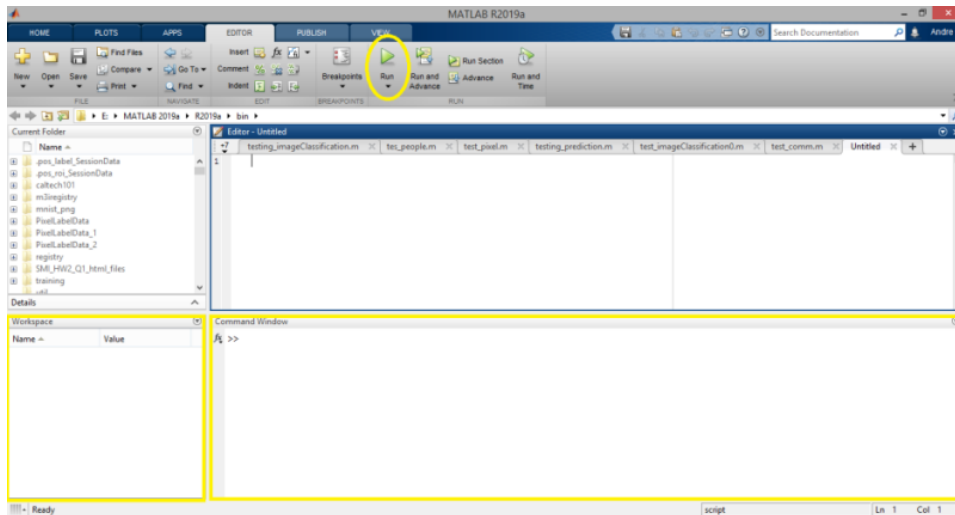


Figure 2.10 MATLAB 2019a UI

To run the function, simply call the function on Command Window or click the Run button in the Editor tab. There are various toolboxes available on MATLAB 2019a. All the toolboxes can be found and run in the Apps tab as shown in Figure 2.11. In addition, all of the called variables can be seen on the Workspace.

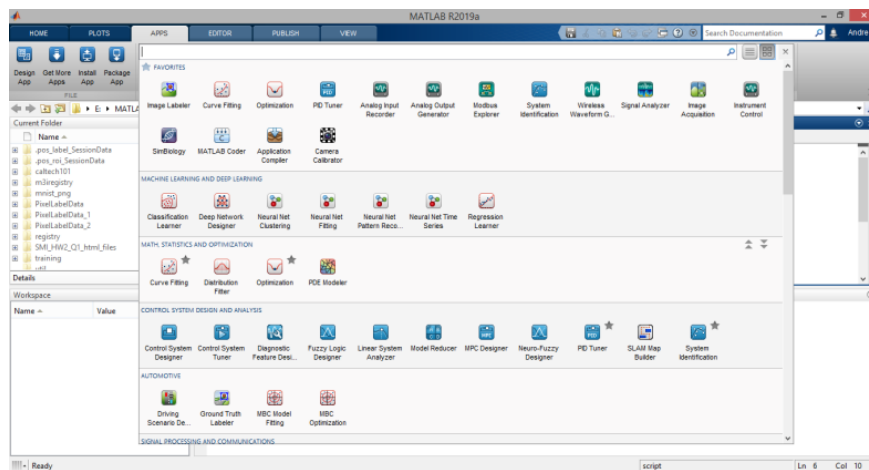


Figure 2.11 Available apps on MATLAB 2019a

There are two toolboxes used in this project which are Machine Learning and Deep Learning also Image Processing and Computer Vision. Furthermore, there are two apps used by the author to integrate the device which are Image Labeler app and Camera Calibrator app.

2.2.1.1. Image labeler app

Image Labeler app is used to define the region of interest of certain object. Click on Load button to upload a bunch of specific pictures. Moreover, press Export button to generate the table of region of interest as shown in Figure 2.12.

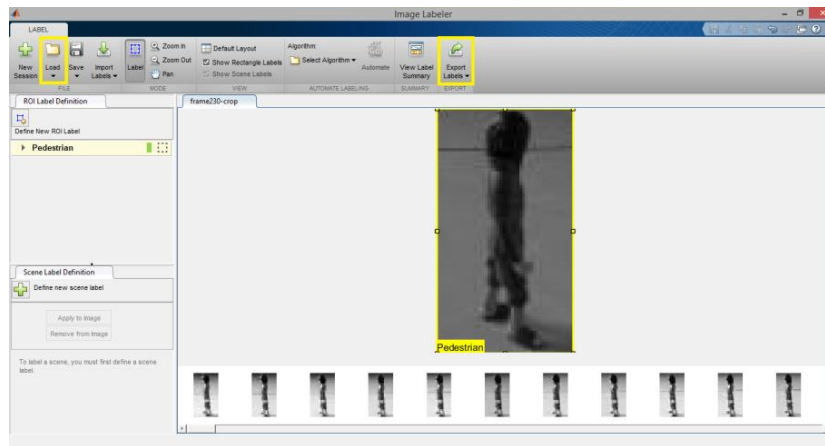


Figure 2.12 Image labeler app UI

2.2.1.2. Camera calibrator app

Camera Calibrator app is used to generate camera extrinsic and intrinsic parameters. It uses check board pattern to estimate camera parameters. Press Add Image button to upload check board pattern images with different angle to the app as shown in Figure 2.13. Furthermore, click the Calibrate button to start the calibration process.

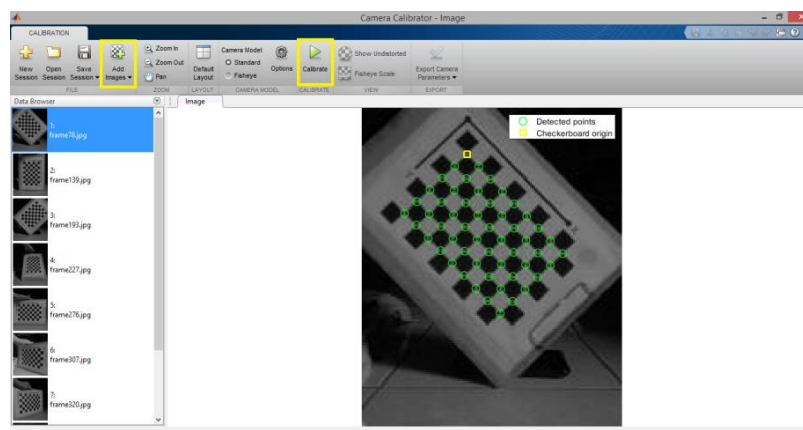


Figure 2.13 Camera calibrator UI

2.2.2. Arduino IDE

Arduino IDE is a compiler and text editor at the same time. It uses to program any type of Arduino board. C and C++ is the basic language to program the board on Arduino IDE.

The UI of Arduino IDE is shown in Figure 2.14. Sketch is a name for a code or program written in Arduino IDE [22].

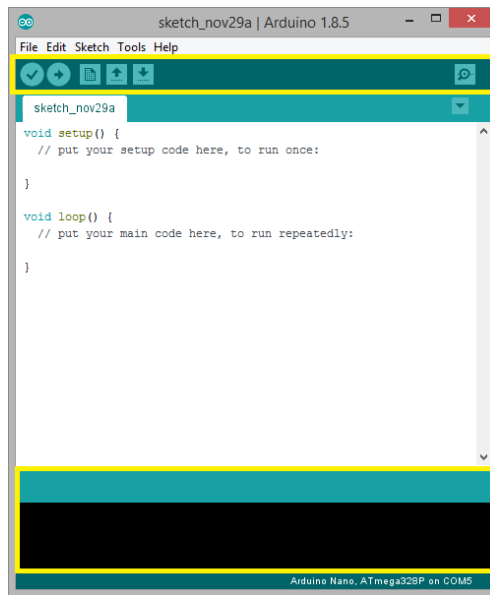


Figure 2.14 Arduino IDE UI

On the top of Arduino IDE there is a toolbar to check, verify, upload, save, and open the code also a serial monitor. On the bottom of Arduino IDE there is a black message box which is basically for displaying status such as error message, compile, and upload program.

2.2.3. IP Webcam android app

IP Webcam app is used to capture images in real-time and connecting PC with mobile phone by using IP address. The UI of IP Webcam app can be seen in Figure 2.15.

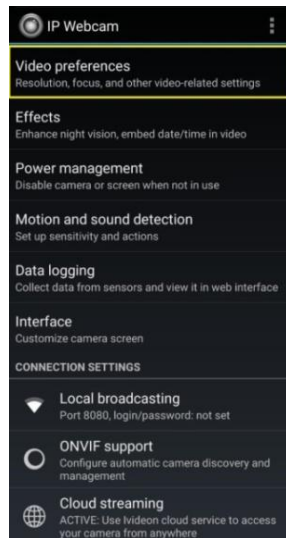


Figure 2.15 IP webcam UI

Press the Video preferences button to see all of the video settings. All of the video setting is changeable as can be seen in Figure 2.16.

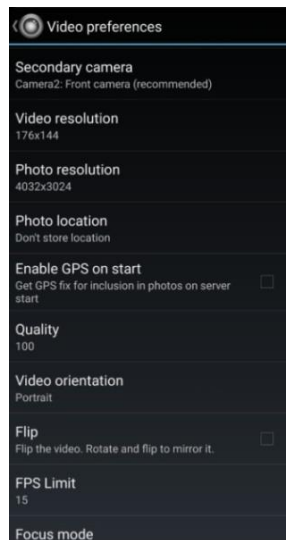


Figure 2.16 IP webcam video preferences

2.3. Hardware

In this final project there are five hardware used. The first one is RC car which basically the foundation of the device. The second one is Arduino Nano to control the remote controller. The third one is mobile phone mounted on the RC car for taking the images through its camera. Next is Dual In-line Package (DIP) switch which used to connect the car controller and the Arduino. The last one is a laptop as the brain of the device.

2.3.1. RC car

The standard 2.4 GHz RC car is used in this project as shown by Figure 2.17. The dimension of the car is 12 cm x 22 cm x 12 cm which represent width x length x height respectively. Note that the remote control is the Single Pole Single Throw (SPST) type button, so it is not the analog one.



Figure 2.17 RC car

The car is powered up by using 2000 mAh 3.6 V that can provide 20 minutes of play time with two hours of charging time. The schematic of controller button is shown in Figure 2.18.

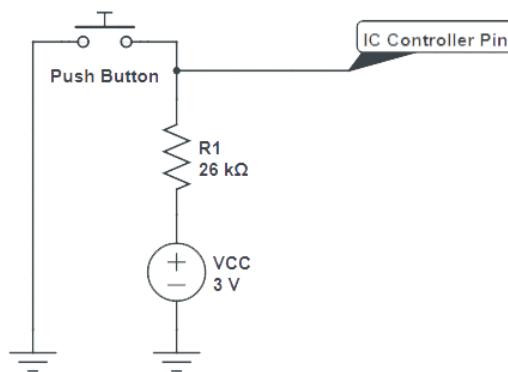


Figure 2.18 Schematic of unmodified controller push button

2.3.2. Mobile phone

Samsung J1 Mini as shown in Figure 2.19 is used in this final project. It fits perfectly on the top of the RC car. Therefore, to mount the mobile phone onto the RC car no screw or tape is needed. Moreover, Samsung J1 Mini's camera has no auto focus feature. It perfect for this project because camera with no auto focus feature is easier to estimate the camera parameters.



Figure 2.19 Samsung J1 mini

Samsung J1 Mini is equipped with 5 Mega Pixel single cameras with $f/2.2$ lens aperture. In addition, Samsung J1 Mini has maximum video resolution of 720 pixels at 30 frames per second. Basically, the aperture of the lens indicates how much light the lens let in. In addition, the focal length of the camera is 3 mm. The focal length measures the distance between the optical center of the lens and the camera's sensor.

2.3.3. Arduino Nano

Arduino Nano as show in Figure 2.20 is a small board compatible with breadboards which based on ATmega328. It has 14 digital pins where six pins can be used as Pulse Width Modulation (PWM) output. In addition, it has eight analog pins, each of which provides 10 bits of resolution (i.e. 0 to 1023).

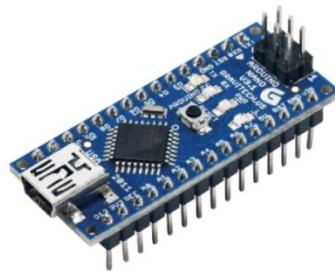


Figure 2.20 Arduino nano board

The Arduino Nano can be powered via the Mini-B Universal Serial Bus (USB) connection, 6-20V unregulated external power supply (pin 30, written as VIN), or 5V regulated external power supply (pin 27, written as 5 V). The power source is automatically selected to the highest voltage source. The specification of Arduino Nano is shown in Table 2.2.

Table 2.2 The Specification of Arduino Nano

Name	Specification
Microcontroller	ATmega328
Operating Voltage	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limit)	6-20 V
Digital I/O Pins	14 (6 of which are PWM)
Analog Input Pins	8
DC Current per I/O Pins	20 mA
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
PCB Size	18 x 45 mm
Weight	7 g

The Arduino Nano has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provide Universal Asynchronous Receiver/Transmitter (UART) Transistor Transistor Logic (TTL) (5V) serial communication, which is available on digital pins 0 (i.e. named RX) and 1 (i.e. named TX). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX light-emitting diodes on the board will flash when data is being transmitted via USB connection to the computer.

2.3.4. Slide DIP switch

The slide DIP switch is used to connect remote controller and the Arduino Nano. It is compatible with bread board. In addition, this project is used six pins slide dip switch as shown in Figure 2.21 to accommodate the entire button on the car controller.



Figure 2.21 Six pins slide DIP switch

To connect between the two joint simply just slide pin connector. Therefore, when the pin is sided into ON state then the two join will be connected. The schematic of the DIP switch is shown by Figure 2.22.

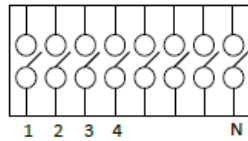


Figure 2.22 Six pins DIP switch schematic

2.3.5. Laptop

Acer Aspire E5-471G is used in this project. It has 4 GB of RAM with Intel i5-4210 processor running on Windows 8.1 Pro. In addition, it has three USB Port 3.0 which support serial communication with the Arduino Nano. The full specification is shown by Table 2.3.

Table 2.3 Acer Aspire E5-471G Specifications

Name	Specification
Processor Type	Core™ i5
Processor Speed	1.70 GHz
Graphics Controller Model	GeForce® 820M
Graphics Memory Capacity	Up to 2 GB
Standard Memory	4GB
Memory Technology	DDR3L SDRAM
Total Hard Drive Capacity	500 GB
Total Number of USB Ports	3
Maximum Power Supply Wattage	65

CHAPTER 3

DESIGN IMPLEMENTATION

3.1. Introductory Remarks

This chapter discusses the system design, hardware implementation, image classification algorithm, training setup and result, and Arduino program. The block diagram that explain the whole system structure and the flowchart that explain the working flow of the device are involved in the system design. Hardware implementation represents the requirement of components and hardware design. All of the requirements to do image classification are described in image classification algorithm. Training setup and result describes all parameters used to train the image classification algorithm. Arduino program describes the program on Arduino to produce motion command.

3.2. System Design

Every 1/30th of a second a mobile phone takes an image and then at certain period the computer grabs the most recent image taken by the smartphone. The computer gets access to the smartphone by using Wi-Fi connection. The computer is running MATLAB 2019a to process each image. After the image is classified by MATLAB, then image class is sent to the Arduino using serial communication via USB port. The Arduino then sends motion command (i.e. stop, turn left, turn right, forward) to the controller based on the image classes obtained from MATLAB. The controller then sends a signal to the receiver in the RC car and makes the RC car moves according to the motion command. The overall system block diagram is shown in Figure 3.1.

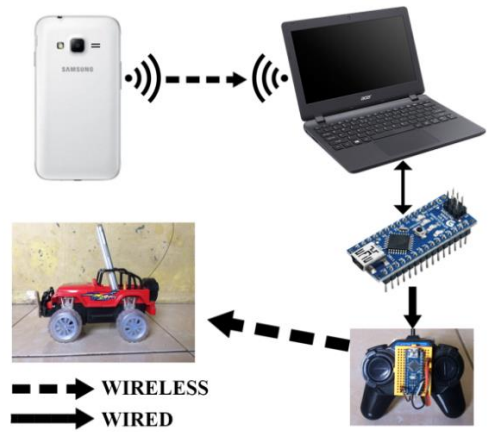


Figure 3.1 Overall system block diagram

The overall flowchart of the system can be seen in Figure 3.2 with the assumption that the distance between the controller and the car is no more than three meters.

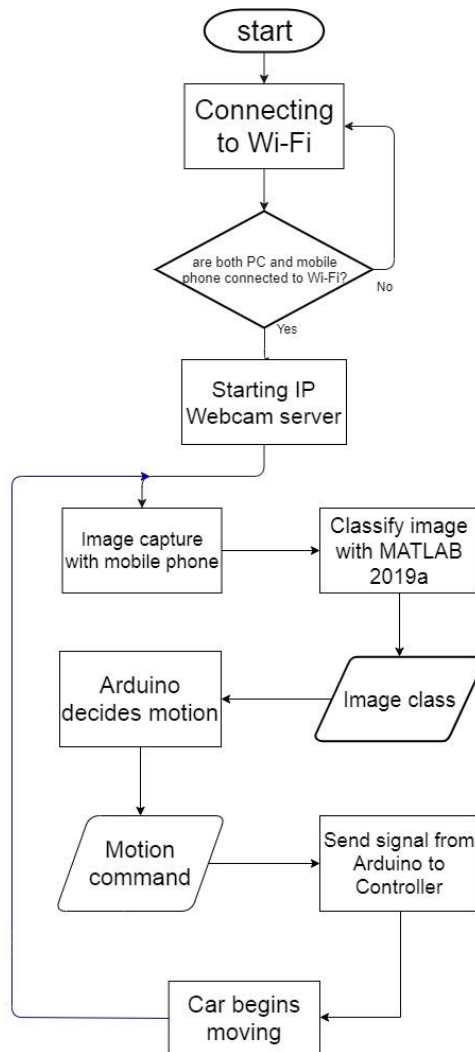


Figure 3.2 Flow chart of autonomous RC car

3.3. Hardware Implementation

All the hardwares that were used to make the device are described in this section. In addition, the modification of controller's circuit is also described in this section.

3.3.1. Car

As mentioned in the earlier chapter, this project is used 1:13 scale RC car with the dimension of 12 cm x 22 cm x 12 cm that represent width, length, and height respectively. The mobile phone is placed on the top of the car as can be seen in Figure 3.3



Figure 3.3 Side looking device

The mobile phone is placed 8 cm from the surface. Notice that there is a slight tilt on the phone of approximately 17° angle relative to vertical axis. Therefore, later on the author will change the value of focal length from the camera calibrator app because the mobile phone placed 90° relative to the surface at the time of calibration.

3.3.2. Controller and Arduino

Arduino Nano is used to send motion command to the controller. Therefore, the controller is modified to recognize the motion command from the Arduino. The unmodified circuit of the controller is shown in Figure 2.18 and the modified circuit of the controller is shown in Figure 3.4.

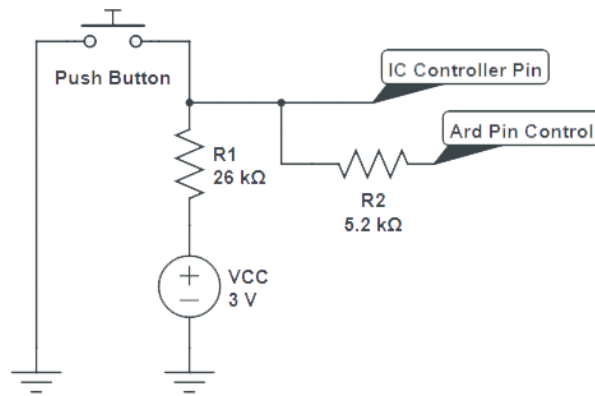


Figure 3.4 Modified controller button

Note that in Figure 3.4, there is only one push button. The controller has four push buttons to control backward, forward, left, and right respectively. Each of push button in the controller is connected to four different Arduino pins. The ground of the controller is connected to the ground of Arduino. The Arduino pin configuration is listed in Table 3.1.

Table 3.1 Arduino Pin Configurations

Pin Number	Details
2	Backward Push Button
7	Forward Push Button
9	Left Push Button
12	Right Push Button
GND	Ground

The controller is operable with the modified circuit while the Arduino is powered on. The controller uses two AA batteries that are connected in series as V_{cc} is equal to 3 V. The output of each Arduino pin is equal to 5 V. Resistors are used to drop Arduino output voltage and limit the current that goes to the controller circuit. It should also be noted that in the modified controller the manual operation is still working. Table 3.2 shows the possible states of push button and Arduino.

Table 3.2 IC Pin Reading

	Arduino Pin HIGH	Arduino Pin LOW
Switch ON	0 V	0 V
Switch OFF	3 V	0 V

The controller is an active low type. So when the IC pin voltage reading is low, it will send signal to the car. When the push button is not pressed and the Arduino pin sending low voltage, the IC pin will read 0.5 V which is still considered as low voltage. Therefore, it will still send the signal to the car. Notice that with the modification in the controller circuit as shown in Figure 3.4 the IC pin would read 0 V when the Arduino is not powered. Therefore, DIP switch is used to cut the connection of the Arduino when manual control is needed (i.e. Arduino power is off). The implementation of dip switch and Arduino are shown in Figure 3.5.

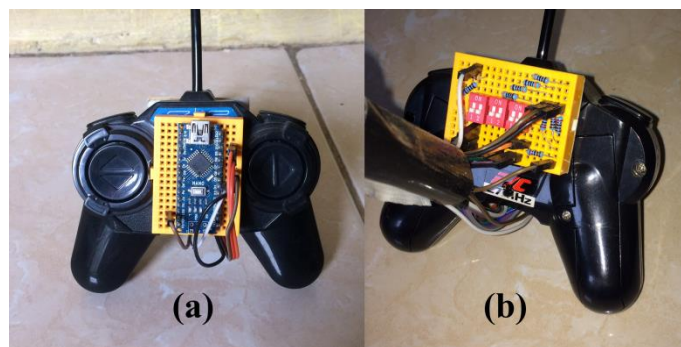


Figure 3.5 Controller as seen from: (a) front side (b) back side

The author used three two-pin DIP switch to accommodate all of the Arduino pins. As mentioned in the earlier chapter, the controller is SPST type switch. So, the controller cannot control the speed of the car. Since the speed of the car cannot be controlled, the author can only use high and low signal from the Arduino to control the movement of the car.

3.4. Image Classification Algorithm

Before applying the image classification algorithm into an image, the image extraction needs to be done. This extraction process involves two parts, IP Webcam and MATLAB. The IP Webcam setup and MATLAB code to grab the images from mobile phone are described in the image extraction subsection. After the extraction process the next process is image classification. The image classification algorithms consist of two main algorithms. First, CNN, which is used as path classification algorithm. All of the CNN setups are described in the path classification subsection. The second one is Haar-like feature classifier, which is used as object classification. All of the Haar-like feature setups are

described in the object classification subsection. In addition, the distance measurement algorithm is also described in this section. The overall process of image classification is shown in Figure 3.6.



Figure 3.6 Image classification process

3.4.1. Image extraction

IP Webcam android app is used as the platform for the computer to take the image from the mobile phone. This is possible because IP Webcam generate an IP address, so the computer can get access to take the image at any time. The video setting of IP Webcam app is shown in Figure 3.7.

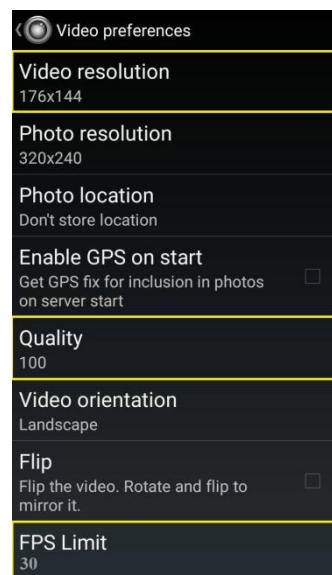


Figure 3.7 Video preferences on IP webcam app

The video resolution corresponds to the height and width of the image frame respectively. This image size is then used as the parameter of input images in the network. 30 frames per second are used as the limit of the mobile phone. After the video is settled up, just start the server of IP Webcam. Then it will generate IP address as shown in Figure 3.8.

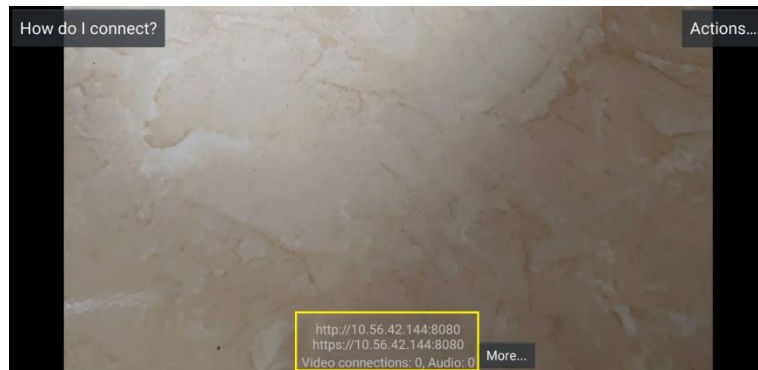


Figure 3.8 Example of IP webcam's IP address

The generated IP address is then used as the variable in MATLAB. The code to grab an image frame is shown in Figure 3.9.

```

Editor - D:\MATLAB\R2019a\bin\ipVideocam.m
ipVideocam.m x train_phase.m x ip_performance.m x test_classification.m x test_pixel.m x
1 - url = 'http://192.168.43.180:8080/shot.jpg';
2 - hVideoIn = vision.VideoPlayer('Name', 'Final Video');
3
4 - folder = fullfile('training', 'NeuralNet');
5 - n = n+1;
6
7 - while (1)
8 -     ss = imread(url);
9 -     ss = colouredToGray(ss);
10
11 -     imageName = strcat('frame', num2str(n), '.jpg');
12 -     rootFolder = fullfile(folder, imageName);
13 -     imwrite(ss, rootFolder);
14
15 -     step(hVideoIn, ss);
16 - end

```

Figure 3.9 Code to grab an image frame from IP webcam

Line 8 of the code is used to grab the image from the IP Webcam's ip address. After the image is taken, it is gray scaled as this project only processes gray scale images. Line 11 is used to name the image, then it is written to the main folder by line 13 of the code. Line 15 of the code is used to display the taken image in real time. Specifically for Haar-like algorithm, cropping image is needed. The setup for cropping the image is explained later in this chapter.

3.4.2. Path classification

There are three classes in the path classification which are straight, left, and right. All of the data from those three classes that are used in the training must be placed in the same main folder. The code to specify the main folder in MATLAB is shown in Figure 3.10.


```

Editor - D:\MATLAB\R2019a\bin\test_classification.m*
+2  train_phase.m  ip_performance.m  test_classification.m*  test_pixel.m  test_imageClassificat
1 -  digitDatasetPath = fullfile('training','NeuralNet','training01');
2 -  imds = imageDatastore(digitDatasetPath, 'IncludeSubfolders',true,...
3     'LabelSource', 'foldernames');
4
5 -  numTrainFiles = 3100;
6 -  [imdsTrain,imdsValidation]=splitEachLabel(imds,numTrainFiles,'randomize');
7

```

Figure 3.10 Code to specify main folder of training data

Line 5 of the code is used to specify the amount of training data. All of the training setup is described later in this chapter. Notice that on line 6 of the code, the training data and validation data are taken randomly. Therefore, the performance of the network at each training is not the same. The random initialization of weight and other parameters on the network also contribute to these results. The code to make CNN in MATLAB is shown in Figure 3.11.

```

Editor - D:\MATLAB\R2019a\bin\test_classification.m
+1  ipVideocam.m  train_phase.m  ip_performance.m  test_classification.m  test_pixel.m
16 -  layers = [
17     imageInputLayer([176 144 1])
18
19     convolution2dLayer(3,8,'Padding','same')
20     batchNormalizationLayer
21     reluLayer
22
23     maxPooling2dLayer(2,'Stride',2)
24
25     convolution2dLayer(3,16,'Padding','same')
26     batchNormalizationLayer
27     reluLayer
28
29     maxPooling2dLayer(2,'Stride',2)
30
31     convolution2dLayer(3,32,'Padding','same')
32     batchNormalizationLayer
33     reluLayer
34
35     fullyConnectedLayer(3)
36     softmaxLayer
37     classificationLayer];

```

Figure 3.11 Code to initialize CNN

ImageInputLayer function is used to specify the width, height, and number of color channel. There are four input parameters in convolutional2dLayer function. The first parameter (3) refers to the filter size. So, each filter is three in width and three in height. The second one is used to specify the number of filters. So, in this project the first convolutional layer has eight filters, the second convolutional layer has 16 filters, and the third convolutional layer has 32 filters. The third and fourth input arguments are used to specify the padding properties. The word “same” means add padding of size calculated by the software at training or prediction time so that the output has the same size as the input when the stride equals 1.

Note that the default value of stride is 1. There are three input parameters in maxPooling2dLayer function. The first one is used to specify the pool size. The second and third input parameters are used to change the stride value to become two from one as the default value. The input argument in fullyConnectedLayer functio is used to specify the amount of classes. This project uses three class (i.e. straight, left, and right) to classify the path. The overall network design is shown in Figure 3.12.

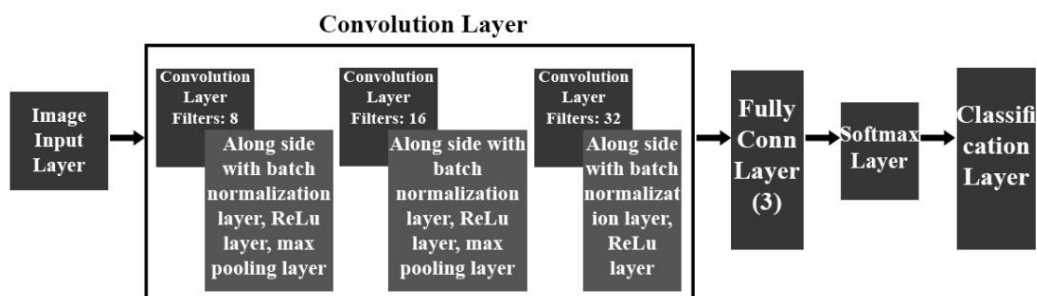


Figure 3.12 CNN design

3.4.3. Object classification and object distance measurement

There are three classes in the object classification which are car, pedestrian, and stop sign. The same as the path classification, all of the training data must be placed in the same main folder. However, specifically for Haar-like features, there are two main folders which are positive image and negative image folder. The code to train the Haar-like features classifier is shown in Figure 3.13.

```

Editor - D:\MATLAB\R2019a\bin\train_phase.m
+1 | ipVideocam.m | train_phase.m | ip_performance.m | test_classification.m | test_pixel.m
1 - positiveFolder = fullfile('training','Object','Pedestrian','Positive');
2 - addpath(positiveFolder);
3
4 - negativeFolder = fullfile('training','Object','Car','Negative');
5 - negativeImages = imageDatastore(negativeFolder);
6
7 - trainCascadeObjectDetector('test_pedestrian.xml', pedestrian_roi,...
8     negativeFolder, 'FalseAlarmRate',0.00000001,'numCascadeStage',15,...
9     'FeatureType','Haar');

```

Figure 3.13 Code to train Haar-like features cascade classifier

To start the training of cascade object detection, just simply call `trainCascadeObjectDetector` function. There are three type of algorithm built in the `trainCascadeObjectDetector` function. The first one is Local Binary Patterns (LBP). The second one is Histogram of Oriented Gradients (HOG). The third one is Haar-like features. The default algorithm is HOG. To access other algorithm just simply type 'FeatureType' in the input parameters of `trainCascadeObjectDetector` function. The first input argument is the Region of Interest (RoI) that generated by the image labeller app.

Haar-like cascade training is done when it reach certain false alarm rate or reach the number of cascade stages. The default value of false alarm rate is 0.1 while the cascade stage is 10. The algorithm to perform object distance measurement is dependent on the performance of the object detection. Therefore, as mentioned earlier, there is only one object for each of categories to train and test the Haar-like algorithm.

Based on equation 3, the distance measurement can be done when the focal length of the camera is known. Camera calibrator app is used to estimate the focal length of the camera in the dimension of pixels. Other than the focal length, the value of height and width of object in the image also in real life should be known. The value of height and width of object in the image can be known by using bounding box of Haar-like algorithm. The dimension of bounding box is in pixels. Since the ratio of width in pixel over height in pixel is always the same as in centimeter, the value of height can be found easily. The code to perform the object distance measurement is shown in Figure 3.14.

```

32 -     bboxCar = step(carDetector,ss);
33 -     bboxPedestrian = step(pedestrianDetector,ss);
34
35 -     valueCar = isempty(bboxCar);
36 -     valuePedestrian = isempty(bboxPedestrian);
37
38 -     if valueCar == 0
39 -         widthPixelCar = bboxCar(1,3);
40 -         heightPixelCar = bboxCar(1,4);
41 -         areaPixelCar = widthPixelCar*heightPixelCar;
42 -         heightRealCar = (widthRealCar*heightPixelCar)/widthPixelCar;
43 -         areaRealCar = widthRealCar*heightRealCar;
44 -         readDistanceCar = focalLength/(sqrt(areaPixelCar/areaRealCar));
45 -     end
46
47 -     if valuePedestrian == 0
48 -         widthPixelPed = bboxPedestrian(1,3);
49 -         heightPixelPed = bboxPedestrian(1,4);
50 -         areaPixelPed = widthPixelPed*heightPixelPed;
51 -         heightRealPed = (widthRealPed*heightPixelPed)/widthPixelPed;
52 -         areaRealPed = widthRealPed*heightRealPed;
53 -         readDistancePed = focalLength/(sqrt(areaPixelPed/areaRealPed));
54 -     end

```

Figure 3.14 Code to estimate object distance

Note that all the focal length and width in centimeter for pedestrian object and car object are initialized in the earlier line of code. The result of the distance is then used as the conditional state of the car (i.e. stop or go).

3.5. Training Setup and Result

This part describes all of the training setup and result of the device. There are two algorithm needed to be trained. The first one is path classification is trained to classify direction of the path (i.e. straight, left, and right). The second one is object classification is trained to detect objects and classify them into each of its class.

3.5.1. Path classification

The training path that is used in this project is shown in Figure 3.15. All of the image classification training data are based on this path. The path is designed using Photoshop CS6.

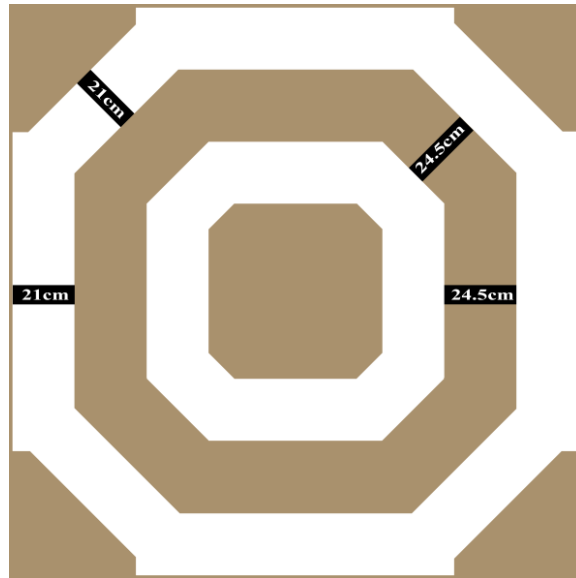


Figure 3.15 Training path

The width of the track is 24.5 cm and the width of the side line is 21 cm. The reason of lane width equal to 24.5 cm is mentioned earlier. 21 cm is chosen as the width of the side line because it is the same as the width of A4 paper. Therefore, later on it is possible to make custom path by only using A4 paper. The circumference of the path is equal to 129 cm. This training path is printed on 195 cm x 195 cm banner. Then, the car is placed on top of it to collect the training data. The data for training is then labeled manually.

The car is set as Figure 3.3 and is driven manually using the controller to take pictures of the path by using the code in Figure 3.8. All of the data is then divided into three different folders named as straight, left, and right. In addition, all of the data should be stored in the same main folder. Notice that the image is gray scaled because the environment colors is not necessary. Furthermore, these also fasten the training time and reduce the processing time of the network. The example of training data and testing result is shown in Table 3.3.

Table 3.3 Example of Training Image and Testing Image

Class	Training			Testing		
Straight						
Left						
Right						

The lowest number of file in those three folders is then used as the maximum number of training data. In this final project, the author used 4300 images from each folder classes as the training data of the network. The rest of the data, which consist of 103 images, are then used to validate the network performance. After the image is taken, then it passes through the CNN. The network setup that is used in this project is shown in Figure 3.12 and listed in Table 3.4.

Table 3.4 CNN Specifications

Layers Name	Size	The Number of Filter	Setting
Image input	[176 144 1]	-	-
Convolutional 2D and (Batch Normalization + ReLu)	[3 3]	8	Padding ("same")
Max Pooling 2D	[2 2]	-	Stride with size 2
Convolutional 2D and (Batch Normalization + ReLu)	[3 3]	16	Padding ("same")
Max Pooling 2D	[2 2]	-	Stride with size 2
Convolutional 2D and (Batch Normalization + ReLu)	[3 3]	32	Padding ("same")
Fully connected	3	-	-
Softmax	1	-	-
Classification	1	-	-

This final project used three consecutive convolutional layers which for each of the first two has one batch normalization layer, ReLu layer, and max pooling layer. The last convolutional layer has additional batch normalization layer and ReLu layer without max pooling layer. After all of the three convolutional layers the image output size is equal to 44 pixel in height and 36 pixel in width from 176 pixel in height and 144 pixel in width. So, one fourth of the image input size is reduced in these three convolutional layers. After that the author attach fully connected layer to the network. Softmax layer and classification layer is the last layer after the fully connected layer. Classification layer is used to generate confidence percentage of the three classes. After the network is initialized, the next step is to train it.

The first step to train the network is to specify the training option of the network. For example, state the number of epoch. One epoch is when an entire dataset is passed through the neural network only once. Too many epochs can lead to over fit the learning result and small epoch can lead to underfit of the learning result. There is no exact numbers of epoch because it different from one dataset to others and it related to how diverse the dataset is. This project used four number of epoch to train the network. Table 3.5 shows the result of the network training in 10 time trials.

Table 3.5 Network Training Results.

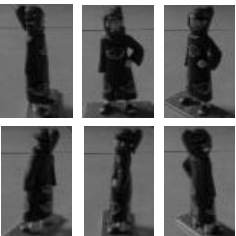

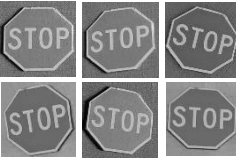
The Number of Trial	Validation Accuracy (%)
1	98.58
2	99.05
3	98.58
4	98.58
5	96.21
6	100
7	97.63
8	98.71
9	98.1
10	99.05

As mentioned before, at each training the weight and other parameters are initialized randomly. Therefore, the validation accuracy and the elapsed time would also differ every training. Based on the data from Table 3.5, the author chose the network with the highest validation accuracy.

3.5.2. Object classification

The data of training is taken manually using the code from Figure 3.9 with car configuration as Figure 3.3. There are three objects used for training (i.e. car, pedestrian, and stop sign). Those three object are scaled down to 1:13 the same as the car. The example of the objects is shown in Table 3.6.

Table 3.6 Example of Objects Training Data

Class	Training
Pedestrian	
Car	
Stop Sign	

All of the 3D objects are taken the images of them from several different orientations. The reason is because there is no way such object recognized in all different position with Haar-like algorithm. Note that every images of object in one category should have almost the same aspect ratio. The aspect ratio, the number of training data, and size for each of the object categories are listed in Table 3.7.

Table 3.7 Training Object Setup

Object	Training Image	Aspect Ratio	Size (W x H x L) (cm)
Car	283	13:14	11 x 5 x 22
Pedestrian	429	16:9	5 x 11.5 x 3
Stop sign	223	1:1	4 x 4 x 0.5

All of the image size from different orientation should rely on the aspect ratio of certain category. To keep this aspect ratio, the author uses Photoscape to crop the image at the desired size. Then, image labeler app is used to export the region of interest in form of table. To make the accuracy of reading object high, this project is only used one object for each category for training and testing. However, with Haar-like algorithm, it is possible to take trained xml file that have diverse data (i.e. pictures of the entire car in certain country) from other researchers and apply it to this project. The training data from each of the categories is placed in different folders. The example of trained object with Haar-like algorithm is shown in Figure 3.16.

```

Automatically setting ObjectTrainingSize to [34, 32]
Using at most 264 of 283 positive samples per stage
Using at most 528 negative samples per stage

--cascadeParams--
Training stage 1 of 15
[.....]
Used 264 positive and 528 negative samples
Time to train stage 1: 61 seconds

Training stage 2 of 15
[.....]
Used 264 positive and 528 negative samples
Time to train stage 2: 77 seconds

Training stage 3 of 15
[.....]
Used 264 positive and 83 negative samples
Time to train stage 3: 42 seconds

Training stage 4 of 15
[.....Warning:
Unable to generate a sufficient number of negative samples for this stage.
Consider reducing the number of stages, reducing the false alarm rate
or adding more negative images.
> In trainCascadeObjectDetector (line 272)
   In train_phase (line 7)

Cannot find enough samples for training.
Training will halt and return cascade detector with 3 stages
Time to train stage 4: 31 seconds

Training complete

```

Figure 3.16 Car training results

Every object category has different number of stages result. In addition, the time to train the network is also different. The number of stage and train time for each object categories are listed in Table 3.8.

Table 3.8 Haar-like Classifier Training Results

Object	The Number of Stage	Training Time (second)
Car	3	211
Pedestrian	2	703
Stop sign	3	321

3.5.3. Object distance measurement

As mentioned earlier, camera calibrator app is used to estimate the focal length of the camera in the dimension of pixels. The result of calibration by using camera calibrator app is shown in Figure 3.17.

```

Command Window
cameraParameters with properties:

Camera Intrinsics
    IntrinsicMatrix: [3×3 double]
    FocalLength: [175.8069 175.6830]
    PrincipalPoint: [69.4431 87.1790]
    Skew: 0
    RadialDistortion: [0.0896 -0.1704]
    TangentialDistortion: [0 0]
    ImageSize: [176 144]

Camera Extrinsics
    RotationMatrices: [3×3×9 double]
    TranslationVectors: [9×3 double]

Accuracy of Estimation
    MeanReprojectionError: 0.0878
    ReprojectionErrors: [54×2×9 double]
    ReprojectedPoints: [54×2×9 double]

Calibration Settings
    NumPatterns: 9
    WorldPoints: [54×2 double]
    WorldUnits: 'millimeters'
    EstimateSkew: 0
    NumRadialDistortionCoefficients: 2
    EstimateTangentialDistortion: 0

```

Figure 3.17 Results of camera calibration

This camera parameter is then stored in the MATLAB workspace. As mentioned earlier, the result of focal length in Figure 3.17 is needed to be tuned. The tuning factor is low because it is only 17° difference in degree. The author is chosen 178.7 as the value of focal length. The different is only 5 pixels from the camera calibrator app result. Other than the focal length, the value of height and width of object in the image also in real life should be known. The value of height and width of object in the image can be known by using bounding box of Haar-like algorithm. The dimension of bounding box is in pixels. The author is chosen arbitrary number for the value of width in centimeter which in this case equal to 8.81 for car object, 8.9 for pedestrian object, and 5.2 for stop sign object. Since the ratio of width in pixel over height in pixel is always the same as in centimeter, the value of height can be found easily. Then, with all of those calculations the value needed to estimate the object distance is collected.

3.6. Arduino Program

After the image is classified by MATLAB, then image class is sent to the Arduino using serial communication via USB port. The Arduino then sends motion command (i.e. stop,

turn left, turn right, forward) to the controller based on the image classes obtained from MATLAB. The Arduino code to receive image class from MATLAB and sends motion command to the controller is shown in Figure 3.18.

```

test_imageClassification0
int value = 3;
int backward = 2;
int gnd = 4;
int forward = 7;
int left = 9;
int right = 12;
int done = 0;

void setup() {
  Serial.begin(9600);
  pinMode(backward, OUTPUT);
  pinMode(forward, OUTPUT);
  pinMode(left, OUTPUT);
  pinMode(right, OUTPUT);
  pinMode(gnd, OUTPUT);

  digitalWrite(backward,HIGH);
  digitalWrite(forward,HIGH);
  digitalWrite(left,HIGH);
  digitalWrite(right,HIGH);
  digitalWrite(gnd,LOW);
}

test_imageClassification0$
void loop() {
  if (Serial.available()>0){
    value = Serial.read();
    done = done+1;

    if (value == 0){
      digitalWrite(left,HIGH);
      digitalWrite(right,HIGH);
      digitalWrite(forward,LOW);
      delay(65);
    }

    else if (value == 1){
      digitalWrite(right,HIGH);
      digitalWrite(left,LOW);
      delay(5);
      digitalWrite(forward,LOW);
      delay(70);
    }

    else if (value == 2){
      digitalWrite(left,HIGH);
      digitalWrite(right,LOW);
      delay(5);
      digitalWrite(forward,LOW);
      delay(70);
    }
    digitalWrite(forward,HIGH);
    Serial.println(done);
  }
}

```

Figure 3.18 Arduino code

The pin initialization is done in the first several lines. In addition, the pin modes for each pin are initialized in the void setup function. The several digital writes in the void setup function is to clear the last motion command. Serial available function is used to check the image class from MATLAB. MATLAB interprets the class of images before sending them to Arduino via serial communication. The image class interpretations are listed in Table 3.9.

Table 3.9 MATLAB Image Class Interpretation

Image Class	Interpretation
Straight	0
Left	1
Right	2

After receive the image class from MATLAB, Arduino sends motion command to the controller based on it. Notice that the forward pin is activated (low signal in this case) only 65 to 70 microseconds. Since the speed of the car cannot be controlled, the author can only use high and low signal also delay from the Arduino to control the movement of the car. At the last of the loop the forward pin is deactivated (high signal in this case) to stop the forward movement. This means that each of the image class that is sent to the Arduino, Arduino sends active motion command for only 65 to 70 microseconds. After the Arduino sends the motion command to the controller, Arduino sends back signal to the MATLAB to let MATLAB know that Arduino is done with its jobs.

CHAPTER 4

RESULT AND DISCUSSIONS

4.1. Results and Discussions

This chapter is divided into four subsections. The first one is the testing of the path classification on the path in Figure 3.15. The second one is the testing of the path classification in the custom track that is made by using A4 papers on carpet. The third one is the testing of the object classification. The last one is the testing of the object distance measurement. Note that the object classification, object distance measurement, and path classification are processed at the same time and in real-time. The division of path classification, object classification, and distance measurement only aims to discuss the entire main feature of the device separately and make it clear. In addition to the four subsections, another section would be added to discuss the weakness of each feature of the device.

4.1.1. Path classification on the training path

The device is tested by operating it on the path in Figure 3.15. This path is made out of a common banner material which is light reflective. As mentioned earlier that the reflected light can be seen as white color relative to the camera. Therefore, it can disturb the path classification process. In addition, the device cannot work when there is a direct sunlight to the path.

Because of the ambient light problem, the testing process is mostly done at night time. Some of the test is done at the day time, however most of the sunlight coming in through the window is blocked by thick curtains and the only light source is from 15 W lamp. The results of the device testing are listed in Table 4.1.

Table 4.1 Device Test Results on Trained Path

Class	The Number of Trial	Succeed	Success rate (%)
Straight	15	15	100.00
Left	15	13	86.67
Right	15	14	93.33

These results are obtained at the optimum lighting condition. Therefore, the path classification process is not disturbed. Analysis on the non-optimum light condition can be found in Chapter 4.1.2.

4.1.2. Path classification on the custom path

The custom path is made out of A4 papers on carpet. Carpet is chosen because it is not light reflective. Therefore, the test can be done at any time. However, although it is not light reflective, an intense direct sunlight might disturb the path classification process because it makes the brightness of images too high. On the other hand, the friction on carpet is relatively bigger than the banner. So, there is a slight modification in the device program. In addition, the friction also makes the device hard to turn left or right. Therefore, the test is done when the device is fully charged.

There are two types of custom path. The first one is the full line as shown in Figure 4.1 (a). The second one is the dash line as shown in Figure 4.1 (b). The distance between each paper in dash line path is 5 cm.

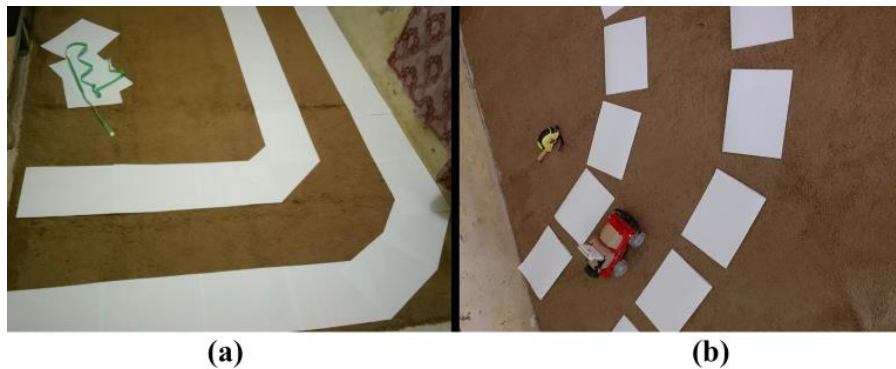




Figure 4.1 Example of custom path on carpet: (a) full line path (b) dash line path

The device testing is divided into two sections. The first one is when there is no direct sunlight to the carpet. The second one is when the sunlight coming in through window is dimmed out using thin curtain. The results of the device testing are listed in Table 4.2.

Table 4.2 Custom Path Testing Results

Path	Condition	Class	The Number of Trial	Succeed	Success rate (%)
	No direct sunlight	Straight	15	15	100.00
		Left	15	13	86.67
		Right	15	14	93.33
	Direct sunlight	Straight	15	9	60.00
		Left	15	8	53.33
		Right	15	10	66.67
	No direct sunlight	Straight	15	14	100.00
		Left	15	13	86.67
		Right	15	13	86.67
	Direct sunlight	Straight	15	7	46.67
		Left	15	6	40.00
		Right	15	9	60.00

The device works best on non-reflective surface when there is no direct sunlight. Note that the success rate of the left class is always below other classes. This happens because the steering wheel of the car is cannot be balanced. The testing result in Table 4.2 is taken when the device is fully charged. Since, the power of the turn left and right is depend on the battery power. When the battery is not full enough, the power to turn left and right can't oppose the resistant force from the carpet. Therefore, the turn left and right is not at the maximum angle.

4.1.3. Object classification

The result of object classification test is taken while the car is moving (i.e. the object classification test is not done separately from path classification). The test is done by placing each class's object at the random place within the path. In addition, the orientation of the object is randomized in between trial. This test is done with 15 trials and the results are recorded. The results of device testing on the object classification are listed in Table 4.3.

Table 4.3 Object Classification Test Results

Class	The Number of Trial	Succeed	Success rate (%)
Pedestrian	15	13	86.67
Car	15	10	66.67
Stop Sign	15	8	53.33

The poor performance of the object classification on car and stop sign is due to its size. The size of the car is relatively big while the view angle of the camera is small. Therefore, it is hard to ensure that the car object is fully covered before the device reaches the object. On the other hand, the stop sign is small and is placed higher than the device. Therefore, the device couldn't recognize the stop sign at a far distance. There is also the problem of false positives (i.e. a negative sample is mistakenly classified as positive). This problem would be covered later in the weakness discussion section.

4.1.4. Object distance measurement

The distance measurement algorithm calculates the distance of the camera parallel to the object. Therefore, the object distance measurement test is done when the object and the device facing each other. In addition, the result of object distance measurement test is taken separately. It means that the result is taken when the device is not operating. The results of object distance measurement testing are listed in Table 4.4.

Table 4.4 Object Distance Measurement Results

Real Distance (cm)	Pedestrian		Car		Stop Sign	
	Estimated Distance (cm)	Absolute Distance Error (cm)	Estimated Distance (cm)	Absolute Distance Error (cm)	Estimated Distance (cm)	Absolute Distance Error (cm)
50	42.85	7.15	46.30	3.70	Undetected	-
45	39.76	5.24	43.73	1.27	Undetected	-
40	38.79	1.21	38.40	1.60	Undetected	-
36	37.87	1.87	37.48	1.48	35.74	0.26
34	34.57	0.57	33.50	0.50	34.42	0.42
30	30.01	0.01	30.87	0.87	Undetected	-
28	29.45	1.45	30.28	2.28	Undetected	-
26	27.90	1.90	28.11	2.11	Undetected	-
Average Error		2.43		1.73		0.34

Notice that the error percentage is not linear. Based on Eq. (1) there are three parameters to calculate the object distance which are focal length, object size in image (i.e. bounding box from Haar-like classifier), object size in real life. There is no problem with the focal length because its value is taken from the camera calibrator app. The problem is related with discrepancy between the object size in image and in real life. The image bounding box sometimes localizes a fraction of pixels from the side of the object which contribute to

change in size of object in real life. This cannot be happen because image size in real-life should be a constant.

That is the reason of the error in distance measurement. In addition, the increasing of the bounding box of object is not linear from distance to distance. This leads to the fluctuation in the error percentage. So to conclude, the result of distance measurement really depends on the object localization performance of the Haar-like classifier. This is also the reason that the distance measurement result of stop sign mostly cannot be done as in this process boundary box often cannot be created as explained in Chapter 4.1.3.

4.2. Device's Frailty Discussions

This section mainly discuss about the weakness of the device based on the testing results. There are two main weaknesses. The first one is the reflected light on the training path. The second one is the high false detection rate.

4.2.1. Reflected light

As mentioned earlier that the path in Figure 3.15 is light reflective. Therefore, the surface of the path is shining when it is illuminated by the ambient light. On the other hand, the carpet surface is not light reflective, but direct sunlight can cause the brightness of images too high and might disturb the path classification process the Figure 4.2 shows the not illuminated and illuminated path at the same device's position.

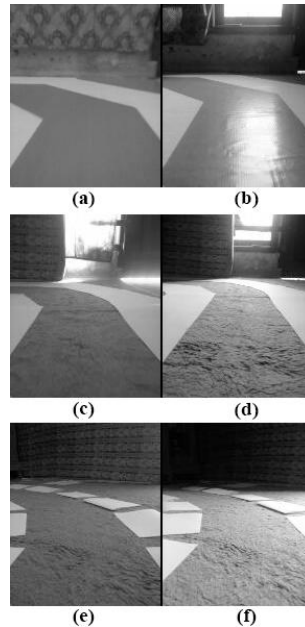


Figure 4.2 Example of path in human eye perspective: (a) not illuminated on training path; (b) illuminated on train path; (c) not illuminated on custom full line path; (d) illuminated on custom full line path; (e) not illuminated on custom dash line path; (f) illuminated on custom dash line path

Figure 4.2 shows some part of the entire testing path when the condition is illuminated and not illuminated by the direct sunlight or ambient light as seen by camera. The effect of the ambient light is clearly shown in Figure 4.2 (a) and (b) even to the human eye. However, Figure 4.2 (d) and (e) seems to be acceptable with respect to the human eye. Figure 4.3 shows the image path with respect to what computer sees.

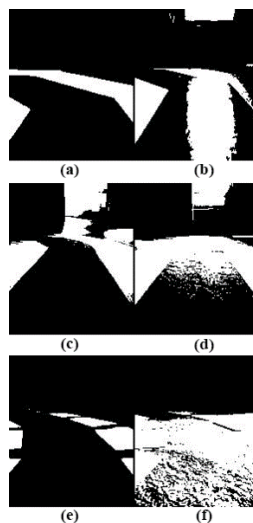


Figure 4.3 Example of path in network's perspective: (a) not illuminated on training path; (b) illuminated on train path; (c) not illuminated on custom full line path; (d) illuminated on custom full line path; (e) not illuminated on custom dash line path; (f) illuminated on custom dash line path

Figure 4.3 is the output of the third convolutional layer and then the result is transformed into the binary image to make it clearer. It is clearly shown that the ambient light or direct sunlight might disturb the path classification process.

4.2.2. False positive

A false positive occurs when a negative sample is mistakenly classified as positive. Haar-like classifier sometimes can be mistakenly classified a really non-identical object. Figure 4.4 shows the example of false positive in pedestrian class. It mistakenly classified a tape as a pedestrian.

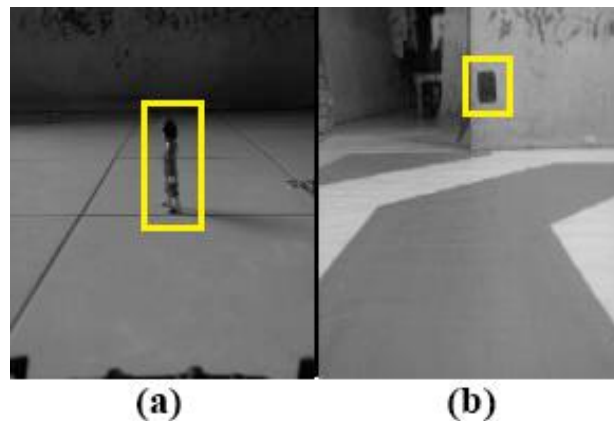


Figure 4.4 Example of false positive: (a) pedestrian object; (b) misclassified object

This might happen because the detector has not enough negative samples or the negative samples are not diverse enough. To perform well Haar-like classifier would need 1000-10.000 images of high quality data (i.e. high diversity rate) [23]. So, to lower the positive rate the author would need to take a background image on every new testing place and re-train the algorithm accordingly. Therefore, from time to time the negative sample images would be varied and the performance would be better. Although the false positive is ever present in the testing process, the performance to detect the true object is considerably good in this case. Because of that the author still uses this algorithm to detect the objects.

CHAPTER 5

CONCLUSIONS AND FUTURE DEVELOPMENTS

5.1. Conclusions

To recapitulate the result and discussions written in Chapter 4 of A DESIGN OF AUTONOMOUS REMOTE CONTROL CAR USING CONVOLUTIONAL NEURAL NETWORK AND HAAR-LIKE FEATURES CLASSIFIER, it can be concluded that:

1. Autonomous RC car is running successfully with the accuracy of the straight class is 97.78%, the left class is 86.67%, and the right class is 91.11%.
2. The object classification is running successfully while the car is moving with the accuracy ranging from 53.33% to 86.67%. There is no program crash throughout the training and test process even though it has three object classes to be processed at the same time.
3. The object distance measurement is successfully implemented with 2.43 cm average of error.

5.2. Future Developments

There are several developments that can be improved from this final project to make the system much better than before. The recommendations for the prototype of this Final Project are:

1. Uses a camera that has a wide field of view. With the wide view angle camera the car object would be easier to detect and the stop sign object can be placed on the side of the road.
2. In order to drive freely in carpet, it is suggested to use a more powerful RC car. The stronger RC car can eliminate the battery problem when running it on the carpet. In addition, it also can eliminate the reflective surface path problem because it working fine on carpet.
3. The object classification can be done by using ANN with the requirement of more powerful computer. The object localization performance can be improved by using ANN and therefore it also improved the object distance measurement performance.

4. The system cannot regulate speed, because the controller is the SPST type. Therefore to be able to control the speed of the car, analog type controller is needed. However, to control the input of the analog controller is another task to be consider. Because the procedure to control digital controller and analog controller is fairly different.
5. Because driving is a continuous task, it is possible to use recurrent neural networks like Long Short-term Memory (LSTM) to capture the sequential relation between neighboring images. Then in situations where the camera is saturated from looking directly in a source of light, the system would use information from previous frames to estimate the decision for the more difficult frame. Changes in speed in the preceding frames could also indicate that the car is approaching a maneuver. The network could learn these relationships and improve its driving.

REFERENCE

- [1] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, New Jersey: Pearson Education, Inc., 2010.
- [2] C. Shulman and N. Bostrom, "How Hard is Artificial Intelligence? Evolutionary Arguments and Selection Effects," *Journal of Consciousness Studies*, vol. 19, no. 7, 8, p. 103–130, 2012.
- [3] W. H. Organization, "Global Status Report on Road Safety 2018," France, 2018.
- [4] E. R. Teoh and D. G. Kidd, "Rage Against the Machine? Google's Self-driving Cars Versus Human Drivers," *Journal of Safety Research*, vol. 63, pp. 57-60, August 2017.
- [5] National Association of City Transportation Officials, *Urban Street Design Guide*, I. Press, Ed., New York: Island Press, 2013.
- [6] M. Bugała, "Algorithms Applied in Autonomous Vehicle Systems," Michał Bugała, Gliwice, 2018.
- [7] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," arXiv, Wales, 2015.
- [8] C. W. Yang, "Autonomous RC Car Control Using Computer Vision," Tunku Abdul Rahman University, Negeri Perak, 2017.
- [9] D. Ungurean, "DeepRCar: an Autonomous Car Model," Czech Technical University, Prague, 2018.
- [10] P. Thomas, *Artificial Intelligence*, United States: Thomson Gale, 2005.

- [11] S. Vieira, W. H. Pinaya and A. Mechelli, "Using Deep Learning to Investigate the Neuroimaging Correlates of Psychiatric and Neurological Disorders: Methods and Applications," *Neuroscience and Biobehavioral Reviews*, vol. 74, no. 4, pp. 58-75, 2017.
- [12] S. Albawi and T. A. Mohammed, "Understanding of a Convolutional Neural Network," in *The International Conference on Engineering & Technology 2017*, Antalya, 2017.
- [13] C. E. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning," arXiv, Glasgow, 2018.
- [14] L. T. H. Phuc, H. Jeon, N. T. N. Truong and J. J. Hak, "Applying the Haar-cascade Algorithm for Detecting Safety Equipment in Safety Management Systems for Multiple Working Environments," *Molecular Diversity Preservation International*, Basel, 2019.
- [15] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Hawaii, 2001.
- [16] E. Baser and Y. Altun, "Detection and Classification of Vehicle in Traffic by Using Haar Cascade Classifier," *International Journal of Advances in Electronics and Computer Science*, vol. 4, no. 2, pp. 137-140, 2017.
- [17] X. Baró and J. Vitrià, "Traffic Sign Detection on Greyscale Image," Open University of Catalonia, Catalonia, 2012.
- [18] A. Joglekar, D. Joshi, R. Khemani, S. Nair and S. Sahare, "Depth Estimation Using Monocular Camera," *International Journal of Computer Science and Information Technologies*, vol. 2, no. 4, pp. 1758-1763, 2011.

- [19] R. C. Gonzalez and R. E. Woods, Digital Image Processing, New Jersey: Prentice-Hall, Inc., 2001.
- [20] P. Alizadeh, "Object Distance Measurement Using a Single Camera for Robotic Applications," Laurentian University, Ontario, 2015.
- [21] Y.-T. Cao, J.-M. Wang, Y.-K. Sun and X.-J. Duan, "Circle Marker Based Distance Measurement Using a Single Camera," Lecture Notes on Software Engineering, Tianjin, 2013.
- [22] S. Monk, Programming Arduino: Getting Started with Sketches, New York: The McGraw-Hill Companies, 2012.
- [23] G. Milner, "Lowering False Positive Detection Rates Using Multiple Haar Classifiers," Florida State University, Tallahassee, 2012.

APPENDIX A

MATLAB 2019a CODE FOR THE DEVICE

Program Code	Description
<code>url = 'http://192.168.43.102:8080/shot.jpg';</code>	Declare the IP address of the camera
<code>% hVideoIn = vision.VideoPlayer('Name','Final Video');</code>	Make video player object (optional, i.e. to see the image from camera)
<code>if ~isempty(instrfind) fclose(instrfind); delete(instrfind); end ardComm = serial('com3','BAUD', 9600); fopen(ardComm); pause(1.5);</code>	Initialization of serial communication between Arduino and MATLAB
<code>carDetector = vision.CascadeObjectDetector('test_car0.xml'); pedestrianDetector = vision.CascadeObjectDetector('test_pedestrian0.xml'); signDetector = vision.CascadeObjectDetector('test_stopsign.xml');</code>	Initialization of cascade object detector for each classes
<code>carDetector.MergeThreshold = 6; pedestrianDetector.MergeThreshold = 6; stopDetector.MergeThreshold = -1;</code>	Initialize the threshold for each object classes
<code>widthRealCar = 8.81; widthRealPed = 8.9; widthRealStop = 8.8; focalLength = 178.7; readDistanceCar = 50; readDistancePed = 50; readDistanceStop = 60;</code>	Initialization of variables for object distance measurement
<code>stop = 0; sign = 0; read = 1; signRead = 10000;</code>	Variable initialization
<code>% folder = fullfile('training','NeuralNet');</code>	Declare root folder (optional, i.e. to write image)
<code>while 1</code>	Code loop
<code>ss = imread(url); ss = colouredToGray(ss);</code>	Read image and turn it into grayscale

Program Code	Description
	image
YPred = classify(net0,ss); YPred = string(YPred);	Path classification
% imageName = strcat(YPred,num2str(n),'.jpg'); % rootFolder = fullfile(folder,imageName); % imwrite(ss,rootFolder);	Save image on the root folder (optional)
bboxCar = step(carDetector,ss); bboxPedestrian = step(pedestrianDetector,ss); bboxStop = step(signDetector,ss);	Object classification
% detectedCar = insertObjectAnnotation(ss,'rectangle',bboxCar,'Car'); % detectedPedestrian = insertObjectAnnotation (detectedCar,'rectangle',bboxPedestrian,'Pedestrian'); % detectedSign = insertObjectAnnotation (detectedPedestrian,'rectangle',bboxStop,'Stop Sign')	Object annotation for each object classes
valueCar = isempty(bboxCar); valuePedestrian = isempty(bboxPedestrian);	Checking the object present for car and pedestrian class
if read ~= signRead+10 valueStopSign = isempty(bboxStop); end	Checking the present of stop sign
if valueCar == 0 % c = c+1; widthPixelCar = bboxCar(1,3); heightPixelCar = bboxCar(1,4); areaPixelCar = widthPixelCar*heightPixelCar; heightRealCar = (widthRealCar*heightPixelCar)/widthPixelCar; areaRealCar = widthRealCar*heightRealCar; readDistanceCar = focalLength/(sqrt(areaPixelCar/areaRealCar)); end	Estimate the distance of car object
if valuePedestrian == 0 % p = p+1; widthPixelPed = bboxPedestrian(1,3); heightPixelPed = bboxPedestrian(1,4); areaPixelPed = widthPixelPed*heightPixelPed; heightRealPed = (widthRealPed*heightPixelPed)/widthPixelPed; areaRealPed = widthRealPed*heightRealPed; readDistancePed = focalLength/(sqrt(areaPixelPed/areaRealPed)); end	Estimate the distance of pedestrian object
if valueStopSign == 0 widthPixelStop = bboxStop(1,3); heightPixelStop = bboxStop(1,4); areaPixelStop = widthPixelStop*heightPixelStop; heightRealStop =	Estimate the distance of stop sign object

Program Code	Description
<pre>(widthRealStop*heightPixelStop)/widthPixelStop; areaRealStop = widthRealStop*heightRealStop; readDistanceStop = focalLength/(sqrt(areaPixelStop/areaRealStop)); end</pre>	
<pre>if readDistanceCar<47 readDistancePed<47 stop = 1; end if readDistanceStop<40 stop = 1; sign = 1; end</pre>	<p>Checking if the object distance break the threshold</p>
<pre>if strcmp(YPred,'Straight') == 1 && stop == 0 fprintf(ardComm, 0); elseif strcmp(YPred,'Left') == 1 && stop == 0 fprintf(ardComm, 1); elseif strcmp(YPred,'Right') == 1 && stop ==0 fprintf(ardComm, 2); end</pre>	<p>Send motion signal to the Arduino through serial communication</p>
<pre>if stop ~= 1 value = fscanf(ardComm); end</pre>	<p>Read signal from Arduino</p>
<pre>if sign == 1 pause(5); sign = 2; signRead = read; end</pre>	<p>Stop if the stop sign break the threshold</p>
<pre>stop = 0; readDistanceCar = 50; readDistancePed = 50; readDistanceStop = 60;</pre>	<p>Re-initialize of variables</p>
<pre>if sign ~= 2 pause(0.015); else pause (0.022); end read = read+1;</pre>	<p>Increment read variable every loop</p>
<pre>% step(hVideoIn,detectedSign); end</pre>	<p>Show the image result (optional)</p>