COMPUTER VISION BASED TRAFFIC SIGNS RECOGNITION SYSTEM

EDWIND LIAW YEE KANG

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2017

COMPUTER VISION-BASED TRAFFIC SIGNS RECOGNITION SYSTEM

EDWIND LIAW YEE KANG

RESEARCH PROJECT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF MECHATRONICS ENGINEERING

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2017

UNIVERSITY OF MALAYA
ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: EDWIND LIAW YEE KANG

Matric No: KQF160002

Name of Degree: Master Degree of Mechatronics Engineering

Title of Research Report: Computer Vision-Based Traffic Signs Recognition System

Field of Study: Computer Vision

I do solemnly and sincerely declare that:

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

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

ABSTRACT

Nowadays, the number of moving vehicles and road users have been increasing very rapidly. Subsequently, more road safety issues have been raised up. Traffic signs on road play a very big role for road safety because it carries important message for the road users especially the drivers. Hence, it is essential that the drivers can notice the traffic signs so that appropriate decision and response during can be made. However, the chances of the drivers overlook some signs are still very high. In order to minimize the said chances, a computer vision based traffic signs detection and recognition system is proposed and developed. The machine learning algorithm, cascaded classifier based on Haar-like features is adopted to develop the traffic signs detection and recognition system. By adopting Haar-like features cascaded classifiers, the traffic signs detection and recognition system with high accuracy is developed.

ABSTRAK

Pada masa kini, bilangan kenderaan bergerak dan pengguna jalan raya semakin meningkat. Oleh itu, semakin banyak isu tentang keselamatan jalan raya telah dipuncakan. Tanda-tanda lalu lintas di jalan raya memainkan peranan yang sangat besar untuk keselamatan jalan raya kerana ia membawa mesej penting bagi pengguna jalan raya terutamanya pemandu. Maka, adalah penting bahawa pemandu dapat melihat tanda-tanda lalu lintas agar keputusan dan tindak balas yang sewajarnya dapat dibuat. Bagaimanapun, kemungkinan pemandu tidak melihat beberapa tanda masih tinggi. Untuk meminimumkan kemungkinan tersebut, sistem pengesanan dan pengiktirafan lalu lintas berasaskan penglihatan komputer dikemukakan dalam kajian ini. Algoritma pembelajaran mesin, pengelasan pengelas berdasarkan ciri-ciri seperti Haar digunakan untuk mengemukankan tanda-tanda lalu lintas dan pengiktirafan sistem. Dengan mengamalkan ciri-ciri seperti Haar mengecil pengelas, tanda-tanda lalu lintas pengesanan dan pengiktirafan sistem dengan ketepatan yang tinggi telah berjaya dikemukankan.

ACKNOWLEDGEMENT

First and foremost, I would like to extend my highest gratitude towards the my advisor, Ir. Dr. Chuah Joon Huang who has given me a golden opportunity to do the research project, study and exploration under his supervision. Dr Chuah has also given numerous valuable comments, suggestions, constructive criticisms and these have certainly helped to improve the quality of this study. His sharing and guidance did not only help me in overcoming the difficulties throughout in this study but also widen my perspective and knowledge in this field.

Secondly, appreciations are to the family members for their understanding and patience without which the efforts are impossible. I acknowledge my sincere indebtedness and gratitude exclusively to my parents for their love, dream, and sacrifice throughout my life.

Lastly, I would like to express my gratitude and appreciations to everyone who has helped me directly or indirectly throughout the study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

CNN	:	Convolutional Neural Network
HOG	:	Histogram of Oriented Gradient
IDE	:	Integrated Development Environment
SIFT	:	Scale Invariant Feature Transform
SVM	:	Support Vector Machine
XML	:	Extensible Markup Language

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

1.1 Introduction

Nowadays, the number of people driving is getting has been increasing. Subsequently, more safety issues have been raised up. Various types of sensing technologies such as GPS, laser rangefinder and even computer vision have been implemented in driving assistance system in order to improve the safety features. One of the most important key to drive safely is being able to notice and watch out the traffic signs on the road which served as warning or awareness to the driver. Even though different kind of colour and shape have been adopted in the design of the traffic sign, the chances of driver overlooking the sign is still high. In this case, it is very essential to have a driver assistance system which can automatically detect and identify different types of traffic signs. By having this, driver can be warned by sounding an audio reminder or giving a warning signals. Moreover, autonomous driving vehicles will be one of the beneficiary from road sign recognition and this is very essential in autonomous navigation.

There are few aspects that making the automatic detection and recognition of traffic signs challenging. Firstly, the types and designs of traffic signs. It is known that traffic signs come with various design and colour. Each type of traffic signs carries its own message. For example, the stop sign at the junction serves the purpose to tell the driver that he should stop his car first before making turn to the left or right. The pedestrian sign is to give alert to the driver that there will be pedestrian crossing the road ahead and there are a lot more. Secondly, the environment surrounding the signs is also an important aspect that has to be considered. The weather conditions and illumination are changing from time to time. Thereby, computer vision has been adopted to address these problems [24, 25, 26, 27, 28, and 29].

Humans use eyes to sense the surrounding world and use brain to compute the information received from the eyes. The science or research that brings the purpose to give a similar or even better ability to a computer or a machine is known as computer vision. Computer vision is often revolving around the topic how a computer or a machine can be made in order to extract and analyze the information from an image or video automatically. Computer vision usually includes development of a theoretical and algorithmic basis to attain automatically extraction and analysis of visual information.

1.2 Problem Statement

In general, traffic signs recognition system is essential for driving assistance system as well as autonomous driving vehicles. However, there are many types of traffic signs, each bringing different information. Hence, traffic signs recognition systems is not only required to detect the presence of traffic signs but to determine what traffic sign it is. A high accuracy traffic sign detection system that can detect and recognize different traffic signs is to be developed.

1.3 Research Objectives

The objectives of this research are addressed as below:

1. To develop traffic recognition system by cascade classifier based on Haar features.
2. To study the accuracy and performance of Haar-based features traffic recognition system.

1.4 Research Scope

The scope of this research project focuses on designing and training a cascade detector for different traffic signs based on Haar features using OpenCV. After training stage, accuracy of trained classifier will be tested. Python Script will also be written and employ the trained classifier to do detection of traffic signs in video feed.

1.5 Thesis Organization

The rest of the thesis is organised as follows:

Chapter 2: Literature Review, presents the background study and review of algorithms for object detection system as well as past research work done on traffic sign recognition system

Chapter 3: Methodology, describes how each step of the research is carried out.

Chapter 4: Results & Discussion, discuss the results obtained and analysis of the results. The strength and weakness of this project are also discussed in this chapter.

Chapter 5: Conclusion, concludes the research findings and states recommendation for future work

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter presents the background study and review of algorithms for object detection system as well as past research work done on traffic sign recognition system.

2.2 Object Detection

Computer vision has been expanding in a very fast pace. Part of the reason is because of the adoption of machine learning approach in this field. Object detection is one of the sub classes of computer vision that has gained a lot of benefits and advancement from the adoption of machine learning methods [2].

Object detection refers to the technique of determining the presence, location and scale of certain object in an image. In other words, the objective of object detection is to determine the presence or absence of a certain class of objects [15]. In many of computer vision application, object detection is first routine to be performed. This is because only after the target object is detected, the following information can be further extracted from the image [2]. For example, in the application of facial recognition, the first detection task must be the task of detecting the presence of human faces. In the review done by reference [2], it also pointed out that object detection has been widely used in different fields, such as human-machine interface (HMI), robotics system, consumer products, security systems, search engines and even transportations.

In the early days, the object detection was done by adopting the techniques of template matching and single part-based models [3]. Later, statistical classifiers or machine learning approach were introduced to object detection. For example, support vector machine (SVM) has been studied and implemented in developing a face detection system in [4]. Other than that, face detection based on neural-network has also been developed in [5]. An example-based learning approach for tracing upright fore views of human faces in complicated scenes is proposed and presented in [6]. 3D object detection has also been proposed in [7]. In reference [7], histogram is adopted to represent various visual attributes and histogram is used as the data set. Reference [8] presented a coarse-to-fine face detection based on SVM. Unlike [3], [8] used coarse-to-fine method to look for faces in image, the processing only concentrates on images containing the positive target object (faces). Face detection based on non-linear SVM is also proposed and presented in [9]. Boosted cascade classifier is implemented in face detection in [1].

Object detection techniques can be grouped into five major types, namely coarse-to-fine and boosted classifier, dictionary based, deformable part-based model, deep learning and Trainable Image Processing Architectures. Each of the types has their strength and weaknesses [2].

2.2.1 Coarse-to-Fine and Boosted Classifier

One of the very famous works in this category is the boosted cascade classifier of proposed in [1]. There are two important keys of the work proposed by [1]. Haar based features were extracted. The second key point of this work is that a classifier of selecting a small number of important features using AdaBoost [11] is constructed. This is because within any image, the total number of Haar-like features is very large, even a lot more than the number of pixels. In order to shorten the time of the classification and make it less computationally expensive, a large majority of the available features needs to be excluded, and emphasize on a small set of critical features. Therefore, if efficiency is the key, coarse-to-fine cascade classifier is the first choice. Another example of the work in this category is the work proposed in [10]. A little modification was done on the traditional Adaboost. In comparison to Adaboost, a backtrack mechanism is used after each iteration of AdaBoost learning in order to reduce the error. Another example of the work is presented in [12] which utilizes boosted classifier to extract haar based features in face detection. Verschae and Ruiz proposed a unified learning framework for detection and classification using a nested cascade of boosted classifier [12]. Figure below shows the block diagram of the unified learning framework that was done by Verschae and Ruiz [12].

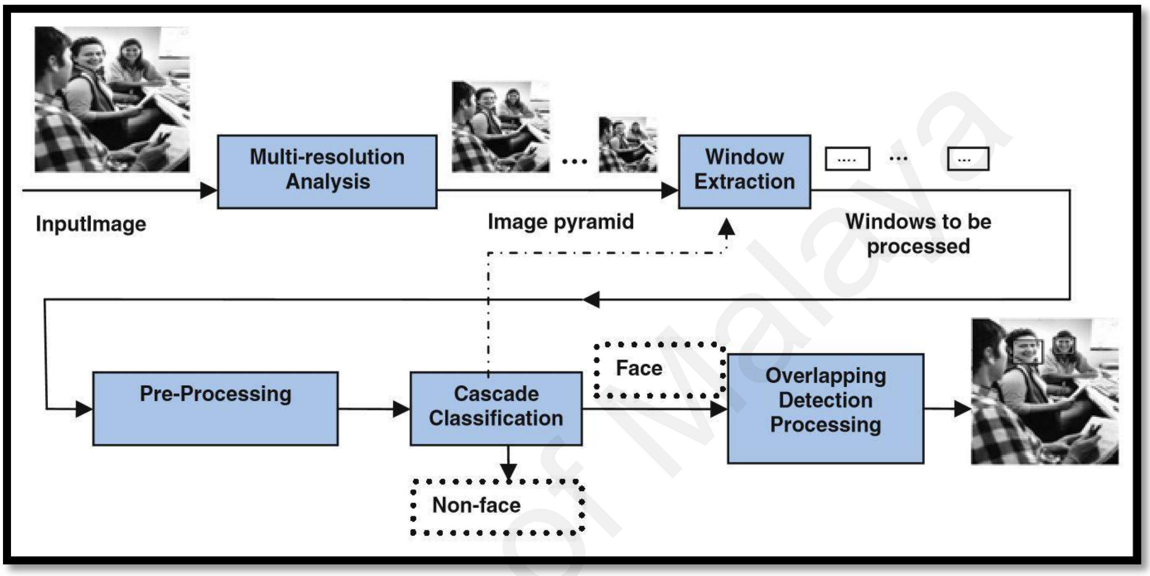


Figure 2.1: Block Diagram of the Unified Learning Framework for Face Detection [15]

2.2.2 Dictionary Learning Based

Dictionary learning based is a technique where elements and features from a dictionary is used to represent objects [16]. One of the drawback of this approach is that it is not suitable to detect multiple object classes in a single image [2]. It means that when more than one object class appear in an image, the classifier can only detect one object class. After removing that object class, the remaining can be determined [17]. An example of this work was the study that done by Mutch and Lowe using this concept for class recognition with limited receptive fields [14].

2.2.3 Deformable Part-Based Model

This technique does not only take object into the consideration but it also considers part models and their relative positions. This approach has higher accuracy in comparison to other approaches but it is more computationally expensive and consumes more time. In reference [18], Felzenszwalb et al. have adopted this approach in developing object recognition system for generic objects such as cars and people. The main challenge they have mentioned is that the objects in such categories can vary greatly in appearance. Variations in illumination and viewpoint may lead to great variation in appearance. In order to solve this problems, the objects to be detected are represented by few parts-based model, as shown in figures below.

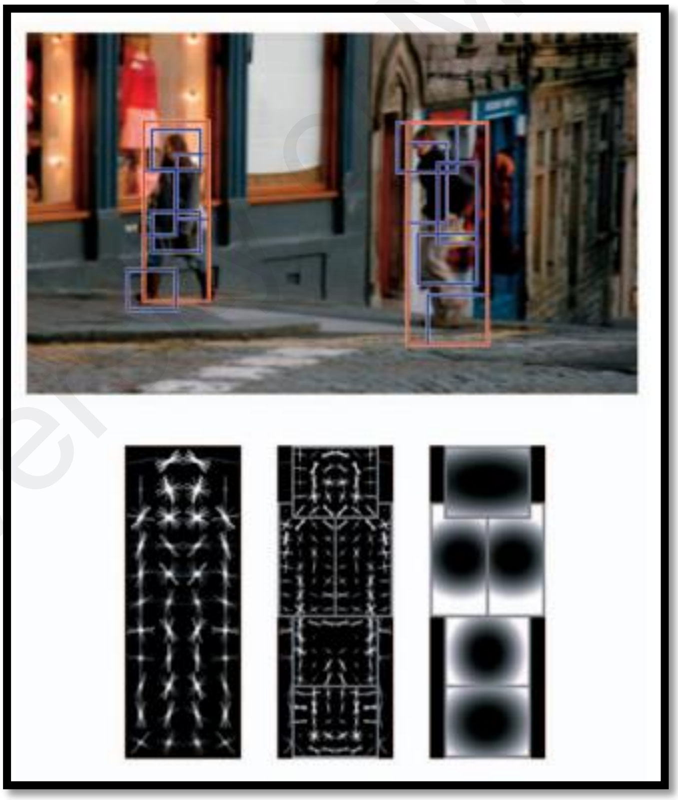


Figure 2.2: Detection of a single person and representation of single person in parts based models [18]

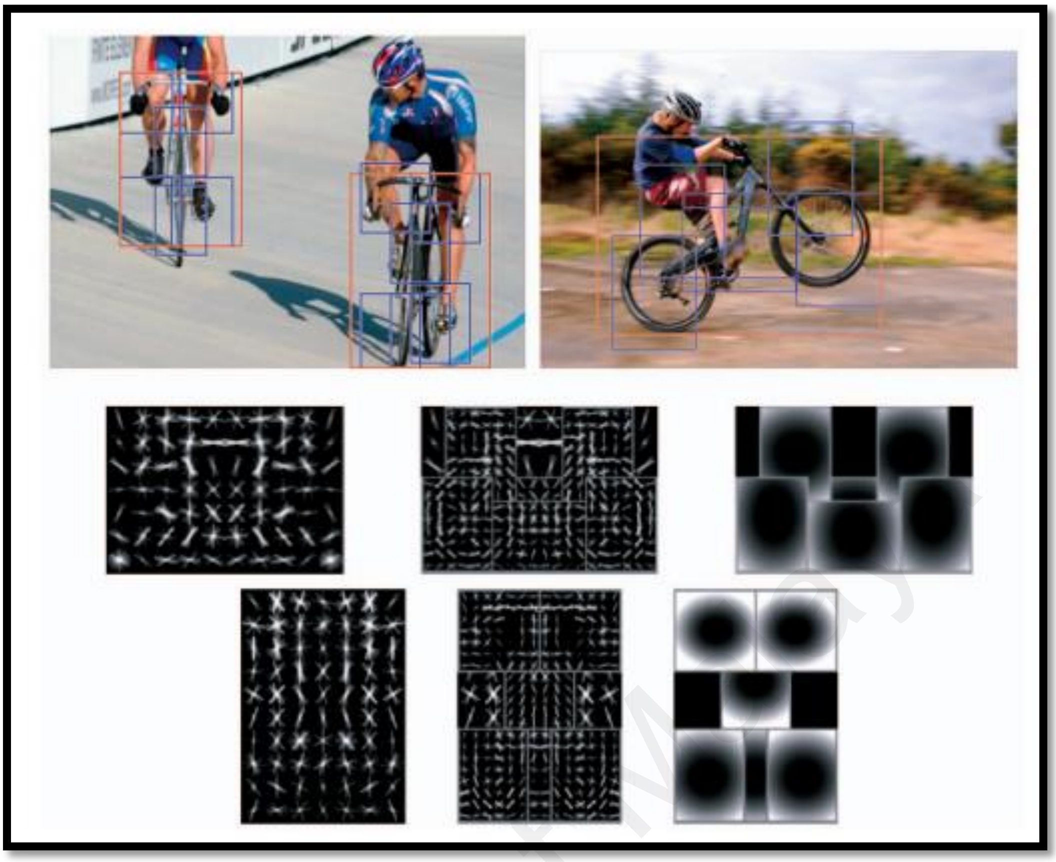


Figure 2.3: Detection of a bicycle and representation of single person in parts based models [18]

2.2.4 Deep Learning

Deep learning is a sub class of machine learning or also known as artificial intelligence [19]. If a well-suited model is designed, this model will be able to solve a complex problem with good accuracy. In accordance to Deng and Yu, deep learning or hierarchy learning is machine learning algorithm that performs the learning task in numerous stages of representation and abstraction [20]. Figure below is the Venn diagram that shows the deep learning in the family of machine learning.

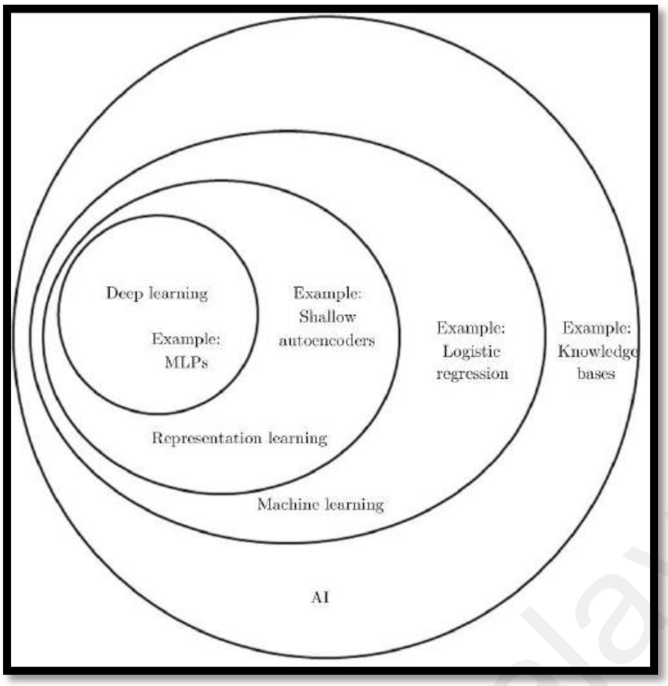


Figure 2.4: Deep learning is a sub set of machine learning [21]

Goodfellow et al has pointed out that Deep learning has relatively higher reliability than other approaches as a machine learning system in real-world condition [21]. Besides, deep learning is representation learning type and it is more flexible in terms of learning because of its higher level of process schematics, which is represented by the figures below.

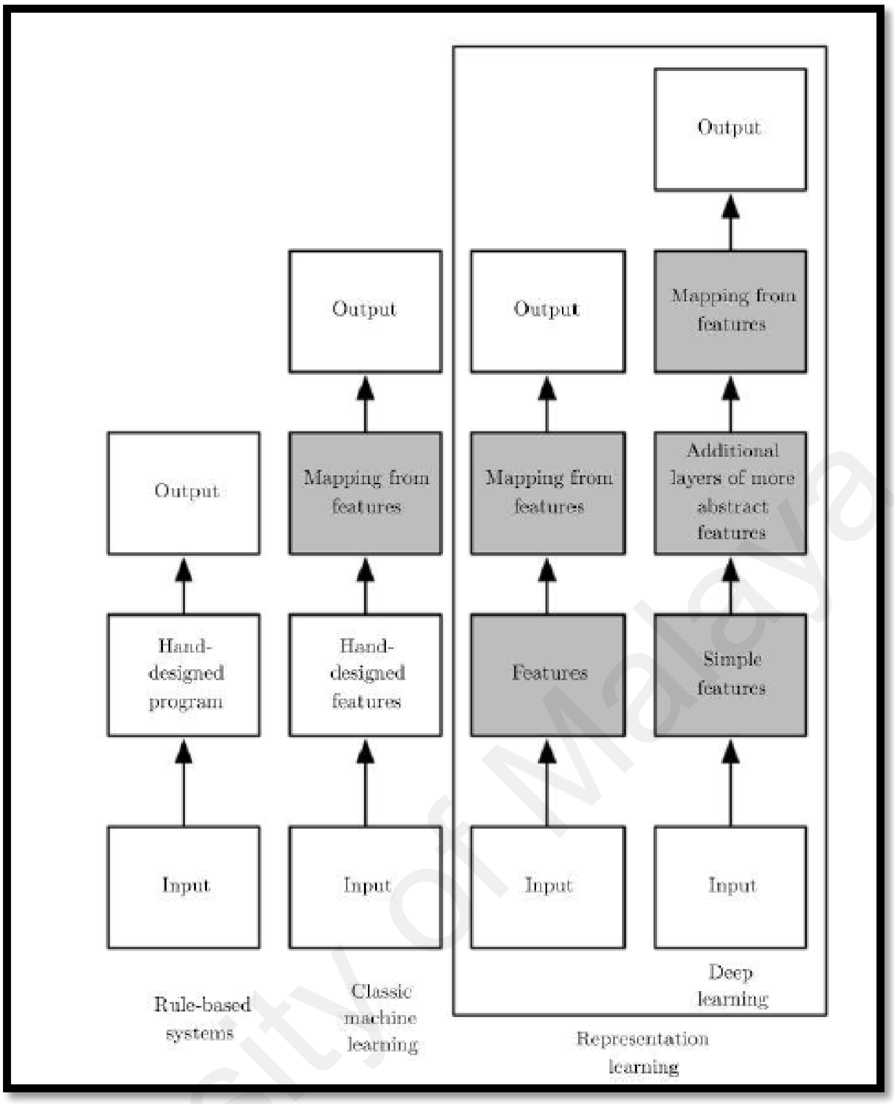


Figure 2.5: Schematic Levels of Each Learning [21]

In comparison with the approaches which have been discussed earlier, the feature representation of deep learning is learned but not designed by the users. The accuracy and reliability of this approach is much higher but the flaw is that it is computationally expensive and requires huge number of training samples [2]. Another example of deep learning is the work presented by Ouyang et al in [22]. A deep model has been proposed. Firstly, the image data was convolved by first filter data map to output features map. Then, the output feature maps were processed by the second layer and the deformation layer. 20 part scores were then output from the results of deformation layer. The deep model is shown in figure below.

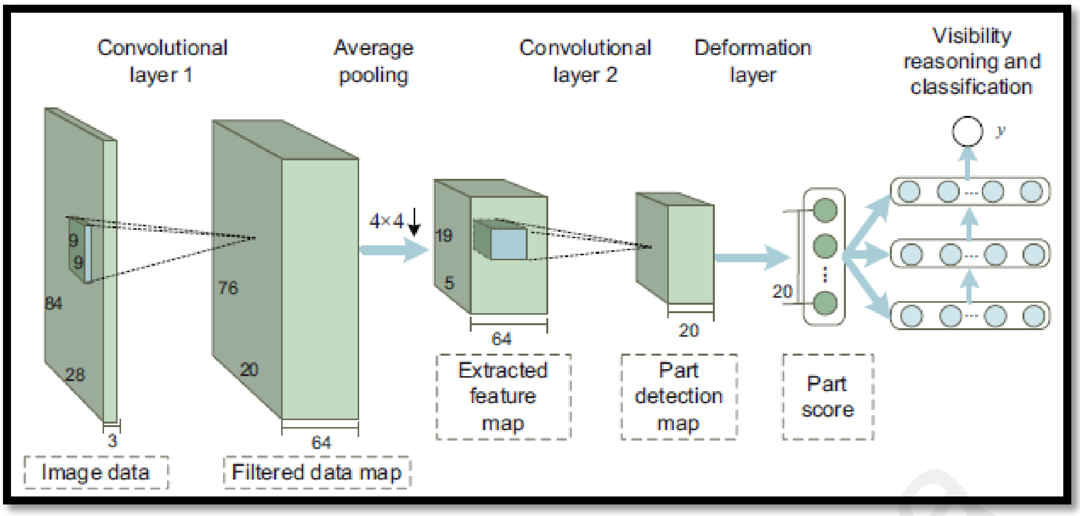


Figure 2.6: Overview of deep model proposed by [22]

2.2.5 Trainable Image Processing Architectures

For this technique, the parameters of predefined operators and amalgamation of operators must be learned first before execution. This approach is a general purpose architecture so it can be used as part of larger system. One good example of this work is the work of Leitner et al. They have designed a humanoid robot by implementing both computer vision and machine learning for the purpose of object [23]. The purpose of implementing this architecture is object identification in 2D plane and further localizing the objects in 3D space plane. Thereby, it requires a few modules to build up this system. Figure below shows the architecture model that proposed by Leitner et al.

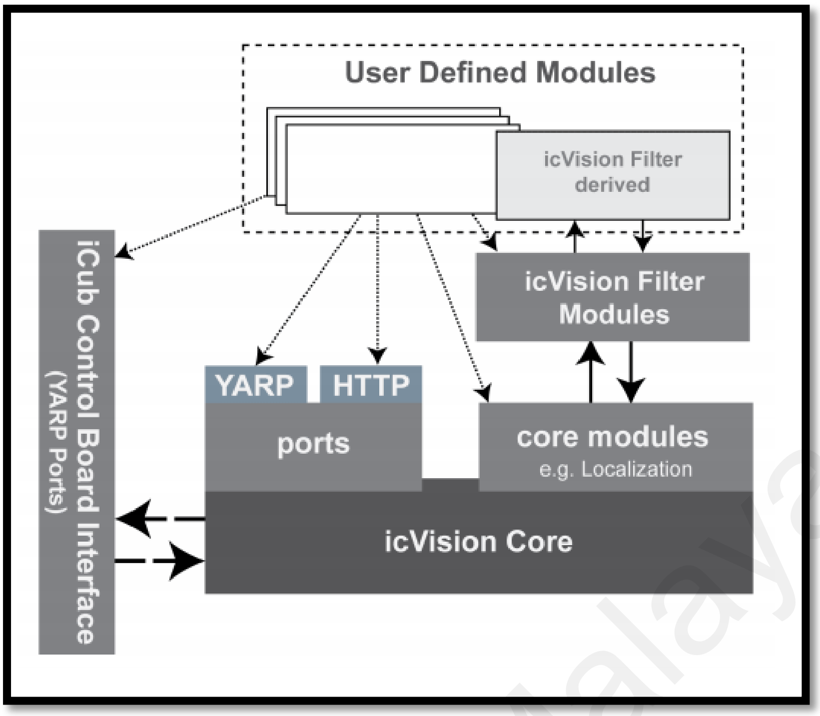


Figure 2.7: Architecture of proposed by Leitner et al. [23]

2.3 Traffic Signs Detection

Shi and Lin have pointed out traffic signs detection rely a lot on colour information [24]. In their opinion, it is not robust to use colour information to detect traffic signs. This is because weather condition, output from different camera, light intensity may result in big variation of colour of the sign. Hence, Shi and Lin have proposed traffic sign detection which does not solely rely on colour information but also the geometry shape of the sign. Their work consists of few steps. Step 1 is to use Histogram of Oriented Gradient (HOG) and linear SVM to determine the regions which have traffic signs. Then, neural network is used to classify the results. Figures below shows the overview of the algorithm proposed by Shi and Lin [24].

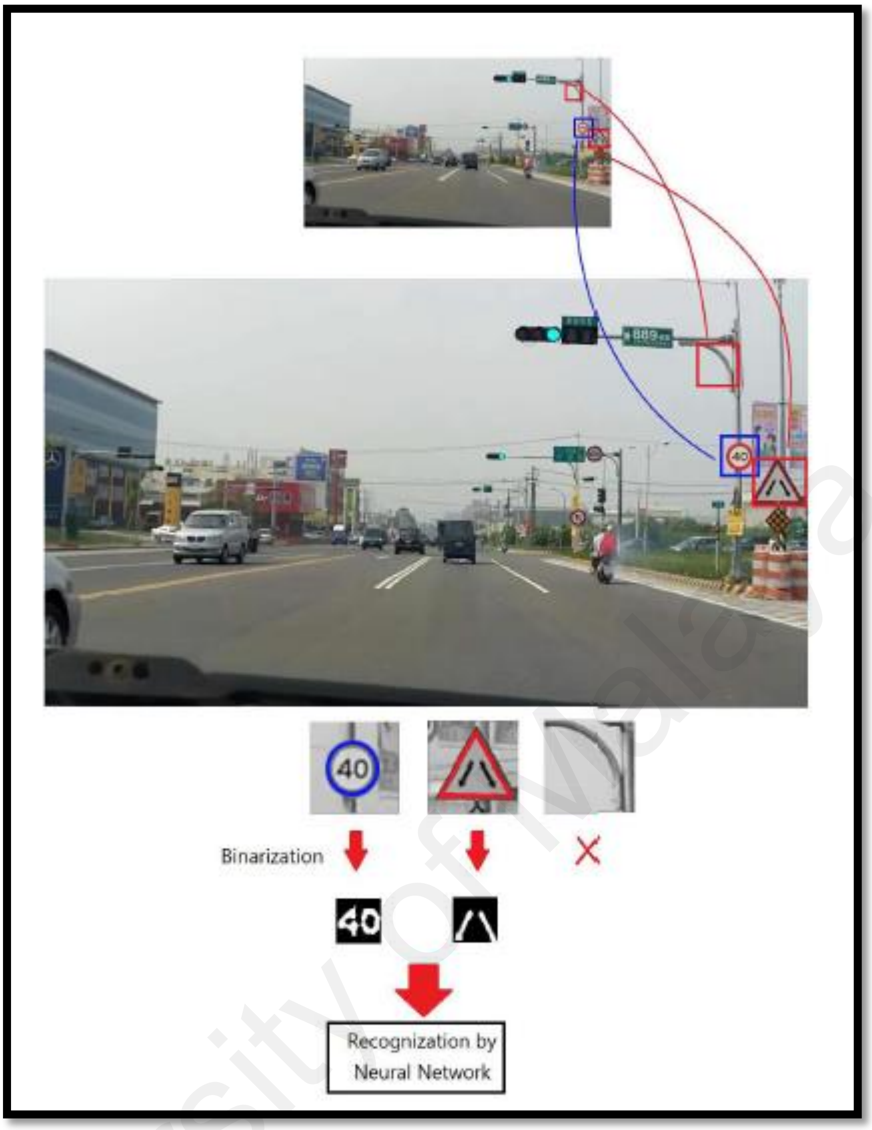


Figure 2.8: Overview of algorithm proposed by Shi and Lin. [24]

As mentioned earlier, Shi and Lin has pointed out the weakness of using colour information alone in doing detection. Figure below shows the comparison of detection between using SVM+HOG and using colour information.

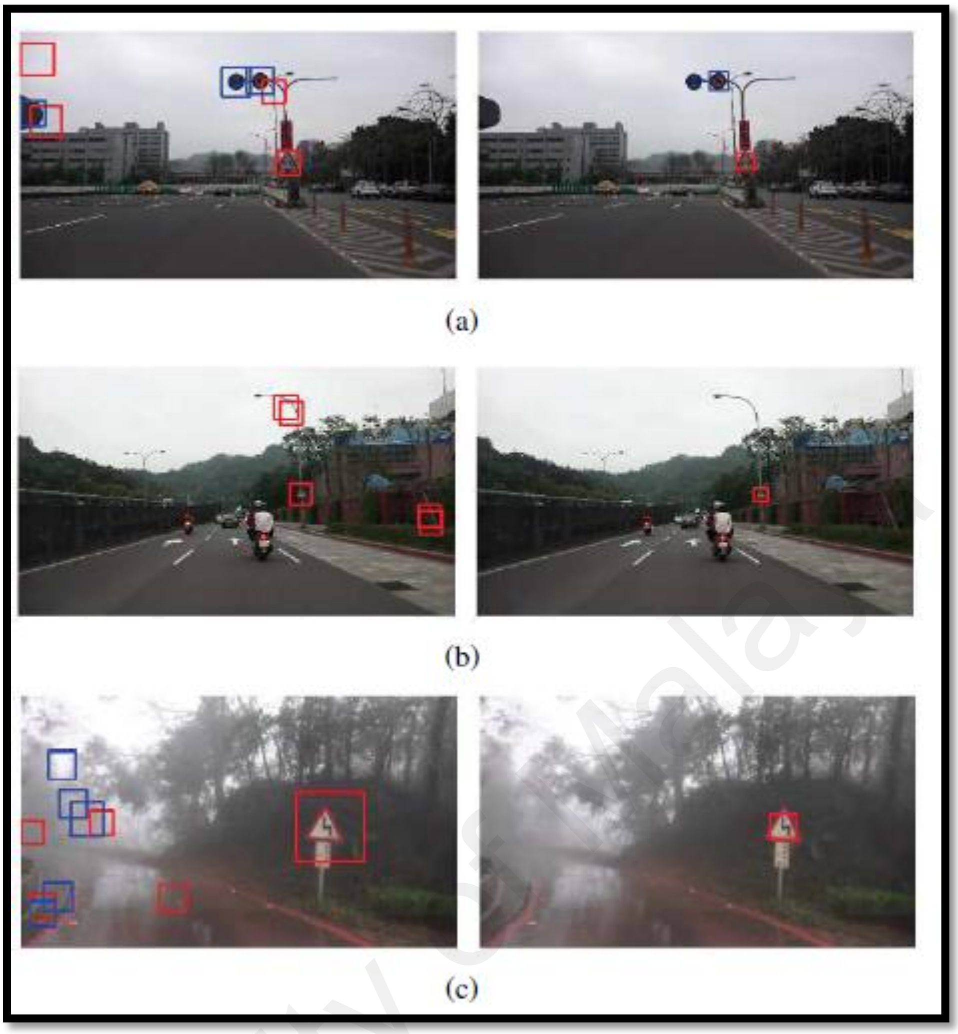


Figure 2.9: Right: HOG+SVM Left: Detection using colour information [24]

From the figure above, it can be clearly seen that in comparison of (a), there are misclassification of signs happened on the right. While traffic signs can be successfully detected in image on the left in (a). From here, it can be concluded that by using colour information alone, it is not robust especially in different weather condition.

Zabihi et al has also proposed the traffic sign detection system. Their detection system is similar to one proposed by Shi and Lin. They also used linear SVM and HOG features in detection stage. For recognition stage, they used colour information and Scale Invariant Feature Transform (SIFT) matching. During recognition stage, the detected images are scaled to same size with template signs. SIFT is then used to do the matching with template signs [25].

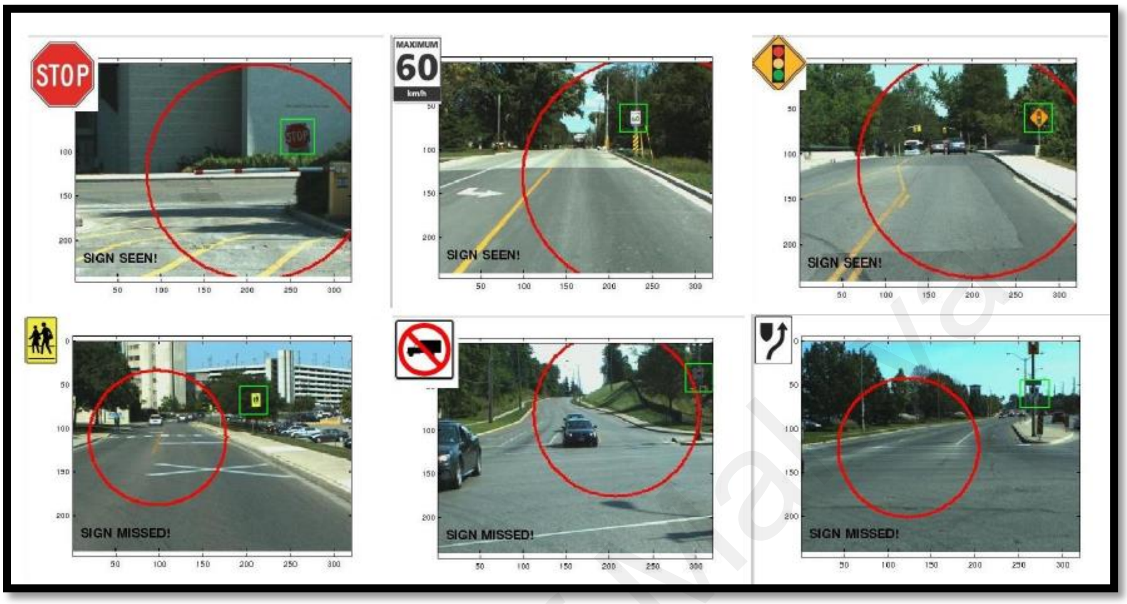


Figure 2.10: Results of proposed system by [25]

Another example of work is presented and described in [26]. In [26], the proposed algorithm has four steps. The first step is candidate regions segmentation. In this step, traffic sign regions are extracted from the environment. The extraction is to ensure the system does not waste time and resource in computing the region that does not contain traffic signs. As traffic signs has different shapes, so the next step is shape classification. Then, feature extraction is the next step. To extract the features, HOG is adopted. Lastly, cascade liner SVM is used in classification and normalization for the purpose of recognizing. The model of cascade SVM is illustrated in figure below.

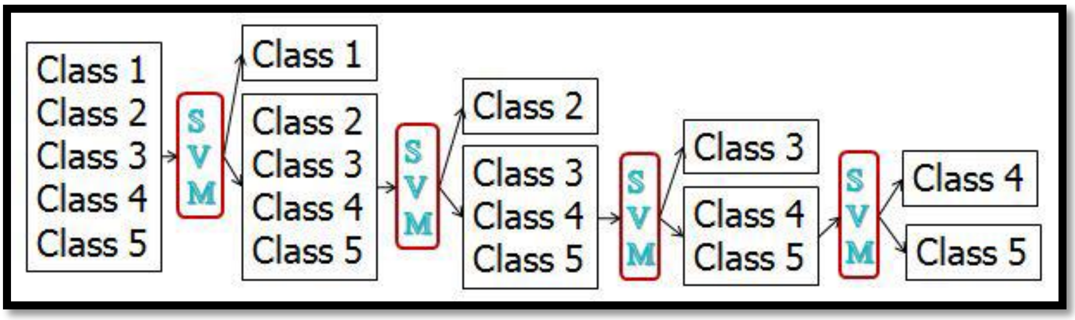


Figure 2.11: Model of cascade SVM proposed by [26]

Karaduman and Eren have proposed deep learning algorithm in determining few traffic signs in [27]. The motivation of the detection and recognition is to determine the driving style of driver. The proposed algorithm is to detect four signs, namely, left and right dangerous curve as well as left and right curve. The convolutional neural network (CNN) has been used to detect the traffic signs. The model of the proposed algorithm is illustrated in figure below.

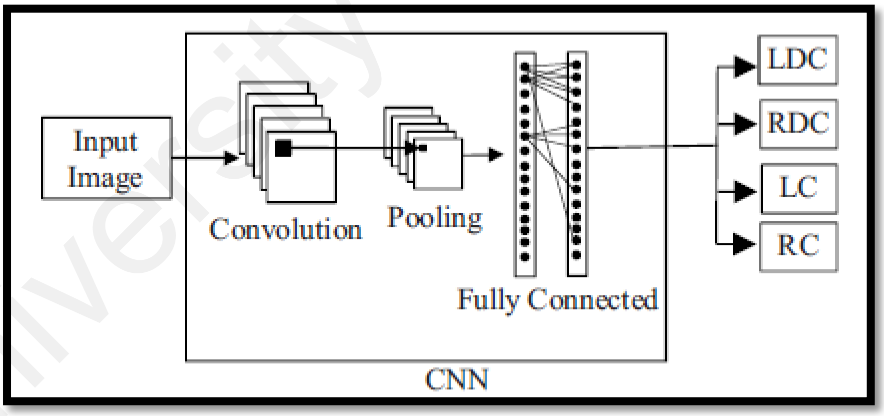


Figure 2.12: CNN model proposed by [27]

Another work of traffic sign detection is presented and described in [29]. The algorithm proposed in [29] consists of 2 main steps. Traffic signs are firstly detected by a set of Haar wavelet features. These features were attained from AdaBoost training beforehand. Second step is to use Bayesian generative modelling for classification. Reference [29] proposed algorithm for traffic signs detection that based on HOG. The colour space adopted are CIELab and YCbCr color spaces. Comparison on the detection of traffic signs based on different type of feature extractions have been done in [27]. The table summarizes the accuracy of traffic signs detection based on different feature extraction method.

Table 2.1: Comparison of Traffic Signs Detection Based on Different Features Extraction

Reference	**Method**	**Accuracy (%)**
[28]	Haar Wavelet	85.00
[29]	HOG	85.00
[27]	CNN	88.02

CHAPTER 3: METHODOLOGY

3.1 Introduction

This chapter covers the discussion of the algorithm adopted to develop traffic sign detection and recognition. The development stage including the training and testing of classifier is covered in this chapter. The development of Python Script is also presented.

3.2 Research Methodology

As per discussed in previous chapter, it is noticeable that most of the traffic sign detection and recognition systems are based on HOG. However, from the comparison that summarized in table 2.1, it is found that Haar like features has similar efficiency compared to HOG and it will take shorter time. For classification, Adaboost cascaded training will be adopted as it takes shorter time to train. In short, it has better efficiency. Unlike what has been done in the pass research work discussed in chapter 2, the method of research that will be adopted in this study is each traffic signs will have its own dependant classifier. For example, stop sign will have its own classifier which can only be used to detect stop signs. After the classifier has been trained, testing will be performed. Once all the classifiers have been trained and tested, a python script will be used to run all these trained classifiers to do detection in video feed from webcam. More details about every step will discussed in following sections of this chapter. The sample signs that chosen are common traffic signs in Malaysia. The signs which will be used for this study are illustrated in figures below.

Figure 3.1: Traffic signs samples used for this study

3.3 Research Flow

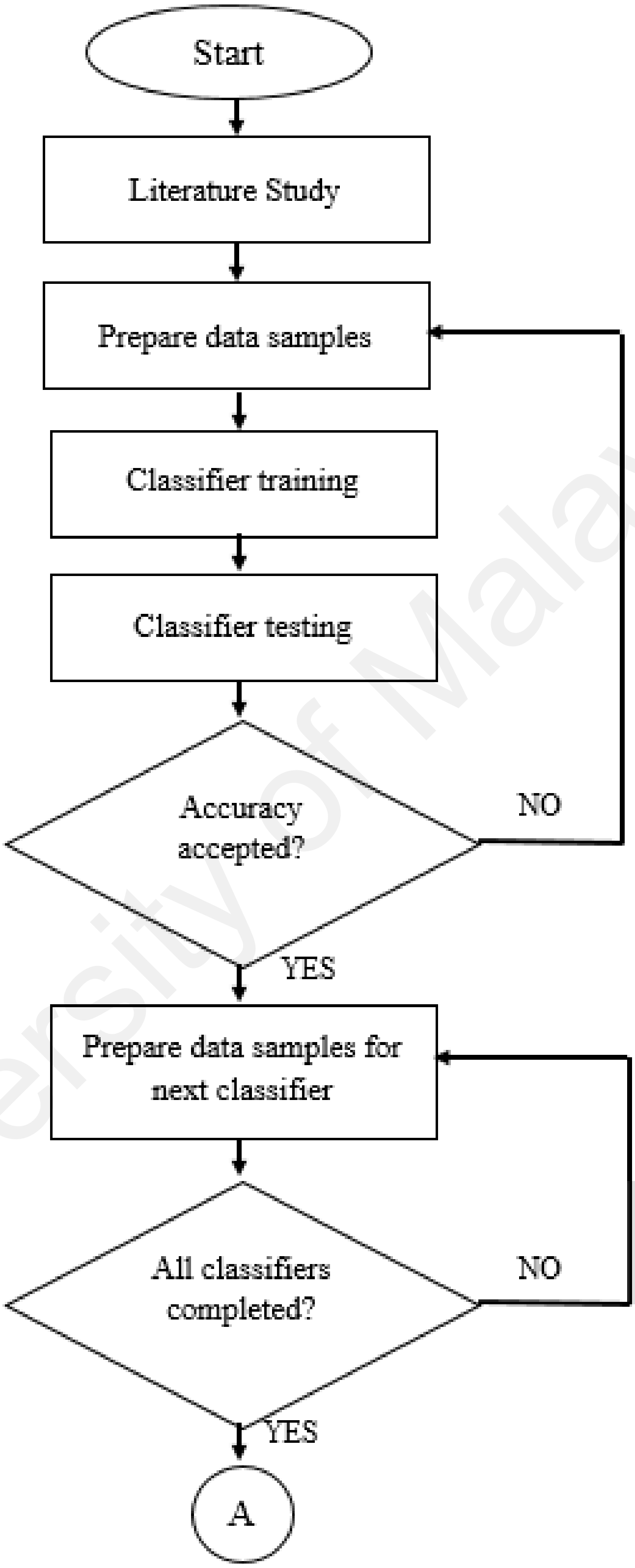


Figure 3.2: Research Flow Chart

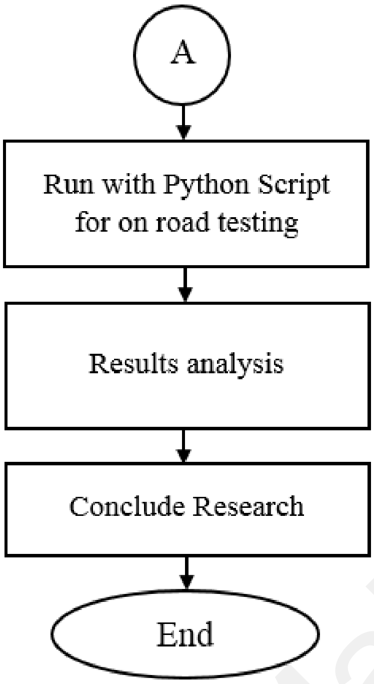


Figure 3.2, Continued: Research Flow Chart

The research flow of this study is presented in figure above. The study was started with literature study in order to make sure the whole research would be carried out in proper track and correct research methods are adopted. Then, it followed by sample data preparation, classifier testing. After all classifier have been trained, the next step is to compose a python script in order to run all the classifier for detection in webcam feed. This testing is to analyse and evaluate the performance when it goes to road condition.

3.4 Research Tool

In this section, the tool used to conduct the research will be discussed in details. Since all the works associated with this research are programming, so the tools for the research are merely few software which have been adopted.

3.4.1 Python

Python IDE (Version 3.5) has been used as an IDE (integrated development environment). The reason of using Python is to have an IDE to develop a script that can integrate the trained classifier into with the real time video feed detection.

3.4.2 OpenCV

OpenCV stands for Open Computer Vision Library. It is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces. It is compatible to run in different operation system, namely Windows, Linux, Mac OS or even phone OS such as iOS and Android. It emphasizes on computational efficiency and real-time applications.

In this research, the version used is OpenCV 3.0.0 and it is used in Windows environment. It was available for download at https://opencv.org/releases.html. Upon finished extracting and installing, the utilities and other tools will be available at the path that it is extracted to. In this research, it is extracted to C drive. All the necessary tools which will be used can be found in the path: opencv-> build-> x64 -> vc12 -> bin. The work folder is shown in figure below.

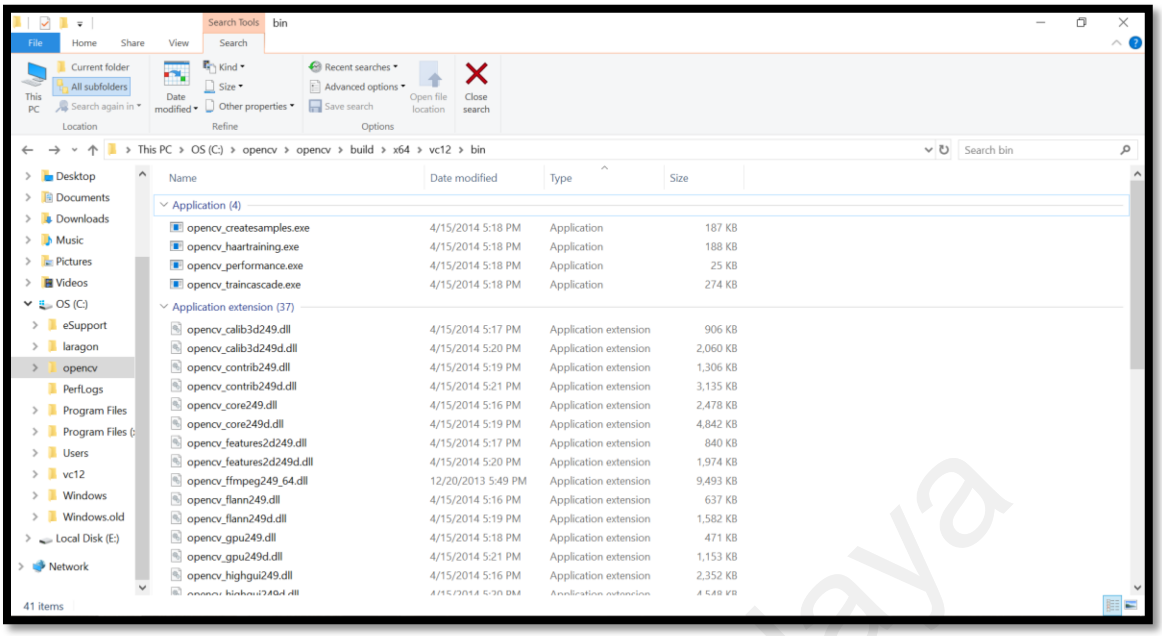


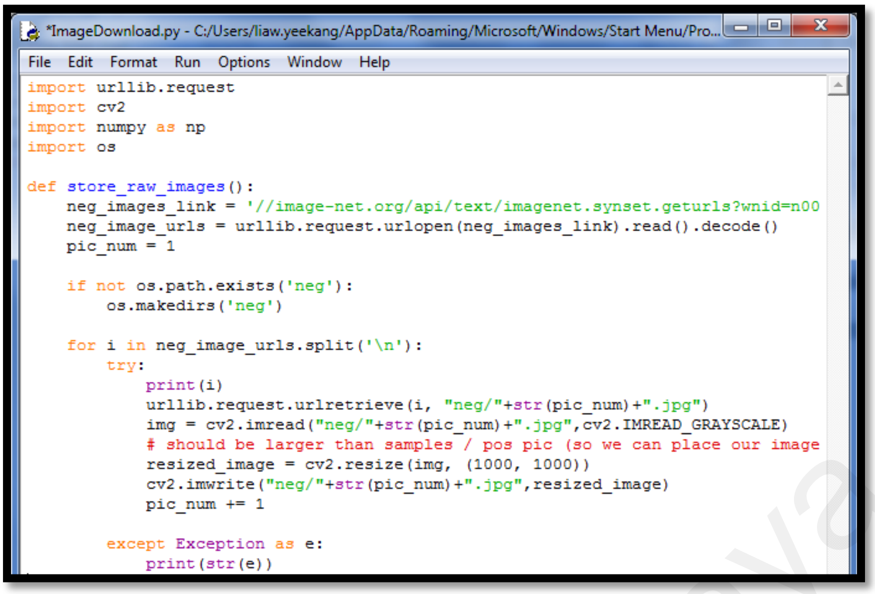
Figure 3.3: Path for OpenCV utility tool

3.5 Data Samples Preperation

To ensure that the accuracy of the trained classifier is high, a large number of data sets must be collected and prepared. The training of the cascade classifiers requires both positive and negative data samples. In this case, positive samples refer to the images containing object that the classifier is required to detect. While negative samples are simply some random images that do not contain any objects to be detected.

3.5.1 Negative Samples

Negative samples are some random images which do not contain object to be detected. A total number of 4,395 images were collected. All these images are converted to grayscale and are resized to the scale of 1000 x 1000 pixels. All the images are sourced from an image database called ImageNet. A Python script in cooperate with OpenCV is adopted to perform this task. Python Script is shown in figure below.



```
*ImageDownload.py - C:/Users/liaw.yeekang/AppData/Roaming/Microsoft/Windows/Start Menu/Pro...
File  Edit  Format  Run  Options  Window  Help
import urllib.request
import cv2
import numpy as np
import os

def store_raw_images():
    neg_images_link = '//image-net.org/api/text/imagenet.synset.geturls?wnid=n00
    neg_image_urls = urllib.request.urlopen(neg_images_link).read().decode()
    pic_num = 1

    if not os.path.exists('neg'):
        os.makedirs('neg')

    for i in neg_image_urls.split('\n'):
        try:
            print(i)
            urllib.request.urlretrieve(i, "neg/"+str(pic_num)+".jpg")
            img = cv2.imread("neg/"+str(pic_num)+".jpg",cv2.IMREAD_GRAYSCALE)
            # should be larger than samples / pos pic (so we can place our image
            resized_image = cv2.resize(img, (1000, 1000))
            cv2.imwrite("neg/"+str(pic_num)+".jpg",resized_image)
            pic_num += 1

        except Exception as e:
            print(str(e))
```

Figure 3.4: Python Script for images download and resize

Once the script is executed, the images will be downloaded into specific directory and images downloaded will be converted to grayscale. On top of that, the images will be resized to scale of 1000 x 1000 pixels. Results are shown in figures below.

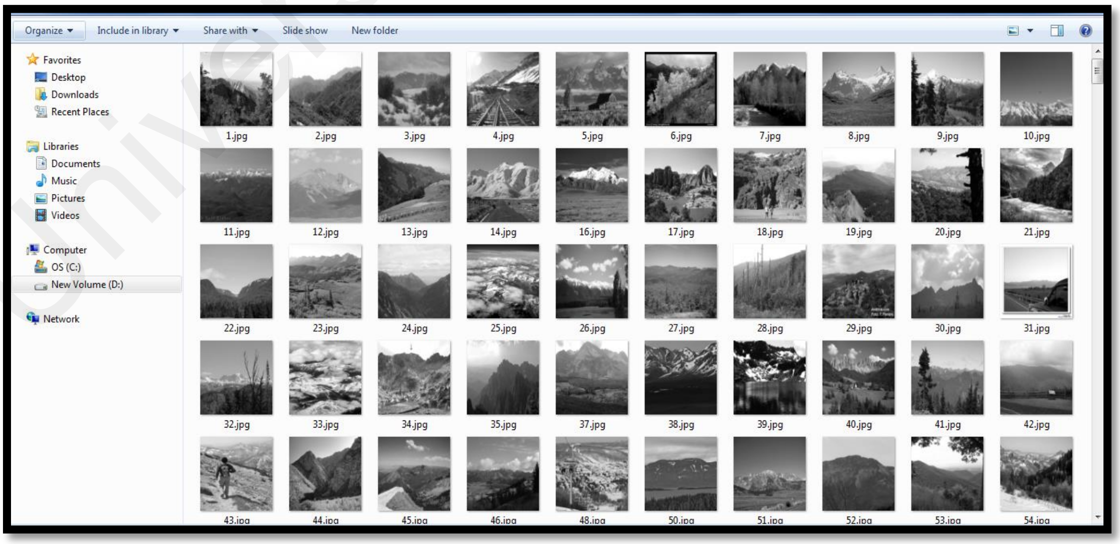


Figure 3.5: Negative Images

3.5.2 Positive Samples

As per discussed, positive samples are the images containing objects to be detected. There are few ways to collect positive samples. For example, to collect the positive images of 60km/h speed limit road sign, one can always go and take pictures of the traffic signs on the road. However, a large number of positive samples are required, so this method will not be considered. To collect and prepare a large number of positive images easily, one of the OpenCV utility is adopted and below steps are taken. In this study, 1 total of 12 traffic signs are to be detected. Thereby, there shall by 12 groups of positive samples. Each group required averagely 4000 images for training purpose. The first step is to collect a few images for respective sign, as shown in figures below.

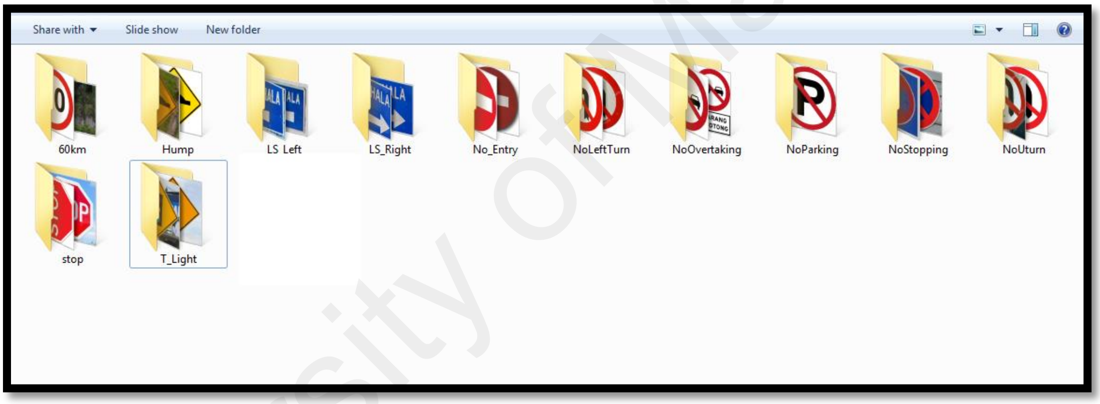


Figure 3.6: Directories of 12 traffic signs

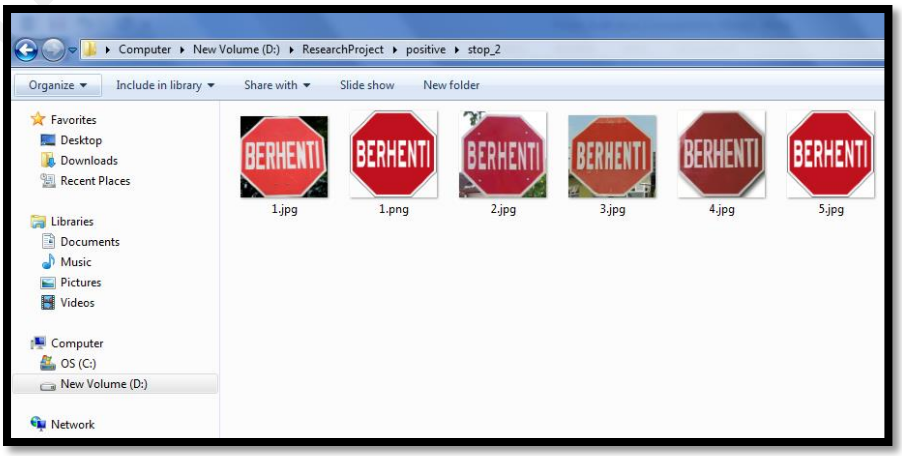


Figure 3.7: Images in each folder

From figure above, images of 12 traffic signs are taken and kept in respective folder. Figure 3.6 shows an example of the images contained in each folder. For example shown in figure 3.6, the stop sign with different background, orientation, illumination are kept in same folder directory. From these images, one of the OpenCV tool, ‘createsample.exe’ is being called. This tool is to create more samples from the original images. What it does is actually randomly infuse the images into the negative images, creating more positive samples. To call and utilize this tool, a batch file is composed, as shown in figure below.

```
createsamples\opencv_createsamples.exe -img positive\Stop_2\1.jpg -bg NewNegative.txt -info Data_Sample\stop_2\2\samples2.lst
-pngoutput info -bgcolor 0 -bgthresh 0 -maxxangle 1.1 -maxyangle 1.1 -maxzangle 0.5 -maxidev 100 -num 4000
```

Figure 3.8: Command in batch file to create more positive samples

The batch file will be executed in command prompt. The batch file and each command has the following meaning.

- img: Source object image directory and name (e.g., a company logo).
- bg: Background description file; contains directory of a list of negative images
- num: Number of positive samples to generate.
- maxidev: Maximum intensity deviation of pixels in foreground samples.
- maxxangle: Maximum rotation angle towards x-axis, must be given in radians.
- maxyangle: Maximum rotation angle towards y-axis, must be given in radians.
- maxzangle: Maximum rotation angle towards z-axis, must be given in radians.

Maximum deviation of pixels is set at 100 and maximum allowed rotation angle towards x, y and z axis of the source object are set at maximum, which are 1.1, 1.1 and 0.5 respectively. These variation in deviation and orientation of angle ensures better accuracy for different scene. Once it is executed, the images will be generated and the results are shown in figures below.

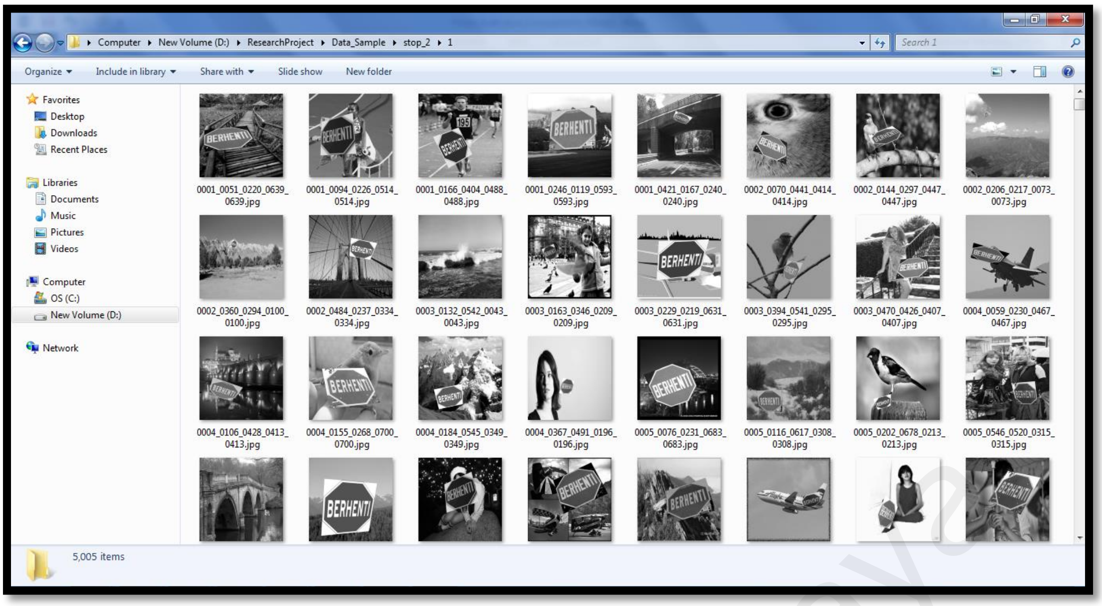


Figure 3.9: Group of newly generated positive images

Figure 3.10: Sample of generated positive images

On top of that, a text file will be generated. This text file contains the name of the generated image and the pixel location of the positive image in that image. This text file will be further resized to 25 x 25 and processed in order to become input for the classifier training, known as vector file. It is known as vector file because it contains the vector of the positive object to be detected by the classifier. The same tool of OpenCV, createsamples.exe is again utilized to process and generate the vector file. To execute the OpenCV createsamples.exe, another batch file is made. The command in the batch file is as follows:

```
createsamples\opencv_createsamples.exe -info Data_Sample\Stop_2\1\samples.lst -num
4000 -w 25 -h 25 -vec positives_Stop_2.vec
```

Figure 3.11: Command in batch file to create vec files

3.6 Cascade Training

'opencv_haartraining.exe' has been adopted to perform the training for the classifier. Similar to the way calling createsamples.exe utility tool, a batch file is first created in order to execute 'opencv_haartraining.exe'. The input for the training are the vector file created earlier and the negative samples. In this study, 4500 positive samples are used and must be in the form of .vec file. For negative samples, 1500 negative samples are used. The command in the batch file to execute 'opencv_haartraining.exe' is shown below.

```
haar\opencv_haartraining.exe -data Train_1\data_stop_2\1\stop -vec
positives_stop2.vec -bg NewNegative.txt -npos 4500 -nneg 1500 -mem 4000 -w 25 -h 25
-nstages 15 -nsplits 30 -mode ALL -nonsym
```

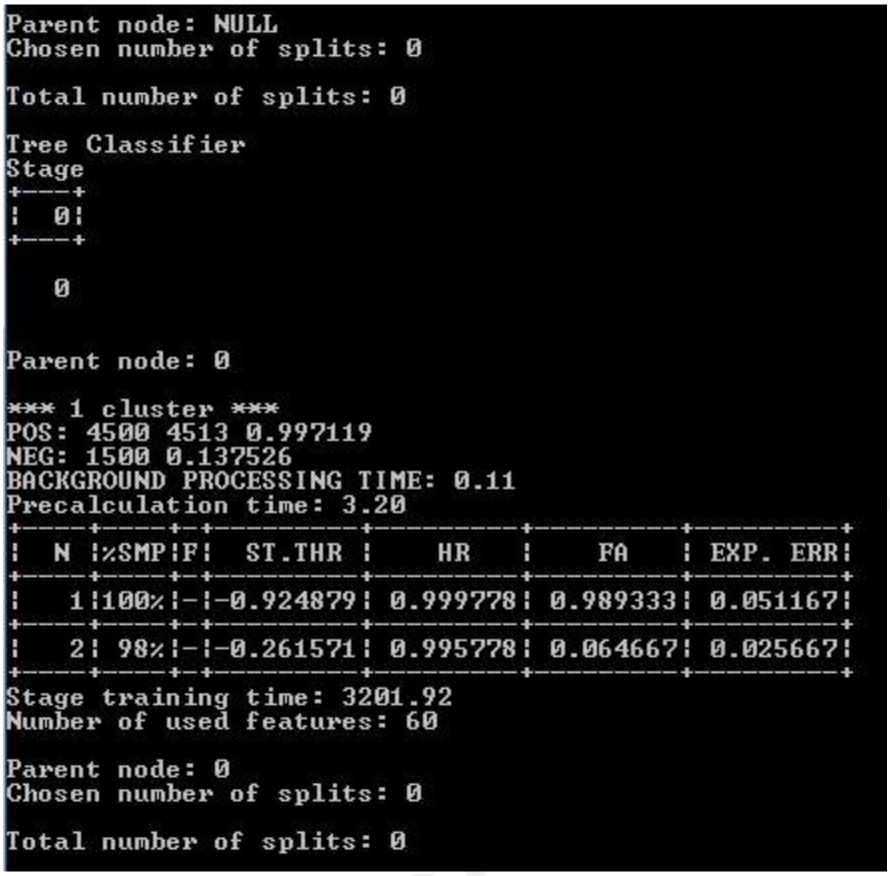
Figure 3.12: Command in batch file for haar cascade training

Each command has the following definition.

- data: Output directory of the .xml file when the training is done
- vec: Vector file that containing all the input file
- bg: Directory pointing the negative images list
- npos: Number of positive samples

- nneg: Number of negative samples
- mem: RAM memory that will be used for training
- w: Width
- h: Height
- nstages: Number of stages

Once the training is started, it cannot be stopped or paused. During the training, the training samples will undergo few stages of training. At each stage, the input will be taken from the output of the previous stage for the training, therefore the number of stages indicating the number of hidden layers in the training stages. Once the training is done, an Extensible Markup Language (XML) file will be created. This file is used as the library sources for the detection system. The training stages of one classifier is shown in the figures below.



```
Parent node: NULL
Chosen number of splits: 0

Total number of splits: 0

Tree Classifier
Stage
+---+
|  0|
+---+

   0

Parent node: 0

*** 1 cluster ***
POS: 4500 4513 0.997119
NEG: 1500 0.137526
BACKGROUND PROCESSING TIME: 0.11
Precalculation time: 3.20
+----+----+-+---------+---------+---------+---------+
|  N |%SMP|F|  ST.THR |    HR   |    FA   | EXP. ERR|
+----+----+-+---------+---------+---------+---------+
|   1|100%|-|-0.924879| 0.999778| 0.989333| 0.051167|
+----+----+-+---------+---------+---------+---------+
|   2| 98%|-|-0.261571| 0.995778| 0.064667| 0.025667|
+----+----+-+---------+---------+---------+---------+
Stage training time: 3201.92
Number of used features: 60

Parent node: 0
Chosen number of splits: 0

Total number of splits: 0
```

Figure 3.13: An example of Completed Training Stage

Based on Figure 3.7, the data display that 0.261571 of threshold stages have been done at the end of this stage with 0.025667 of strong classification error. The time taken for this stage is 3201.92 seconds and the number of features used is 60. The more features it uses, the longer time it takes. Too few of features may lead to failure in training due to inability of the training result to converge. The results of training is summarized in table below.

Table 3.1: Training summary for each classifier

Traffic signs	Data size		Number of stages (Epoch)	Average training time (minutes)	Number of features used
	Positive	Negative			
Stop	4500	1500	11	583	60
One way (left)	4000	2000	6	340	35
One way (right)	4000	2000	5	280	40
Bump ahead	4000	2000	6	200	40
Traffic light ahead	4000	2000	5	350	50
No Overtaking	4000	2000	6	375	40
No Stopping	4000	2000	6	310	60
No U-turn	4500	1500	9	487.4	60
No Parking	4000	2000	4	245	40
No Entry	4000	2000	5	275	40
No Left Turn	4800	1500	6	315.50	30
60km/h speed limit	4500	1500	5	295	60

3.7 Cascade Testing

Classifier must be tested before it can proceed to next step. This stage consists of 2 main steps. Firstly, to prepare and collect for test samples. This is done by utilizing the 'opencv_createsamples.exe' tool. The next step is to do the testing. To test the performance or accuracy of the classifier, another utility tool from OpenCV, 'opencv_performance.exe' is used. Details of each step will be discussed in following section.

3.7.1 Test Data Preparation

Similar to previous step, a batch file is created as shown in figure below. The example shown in batch file below is to prepare the test samples for the testing of one way (left) traffic sign.

```
createsamples\opencv_createsamples.exe -img positive\LS_Left\2.jpg -num 1000 -bg
NewNegative.txt -info Testing\LS_Left\TestSamples\test.dat -maxxangle 0.6
-maxyangle 0 -maxzangle 0.3 -maxidev 100 -bgcolor 0 -bgthresh 0
```

Figure 3.14: Batch file for test sample collection

The batch file above is to create samples to test the performance of one of the traffic sign classifier. The number of the test samples is set at 1000, meaning that 1000 images with that particular sign will be created. Furthermore, a dat file will be created. This dat file containing the pixel location of the object to be detected in each created images. Figure below illustrates the results of the execution of above batch file.

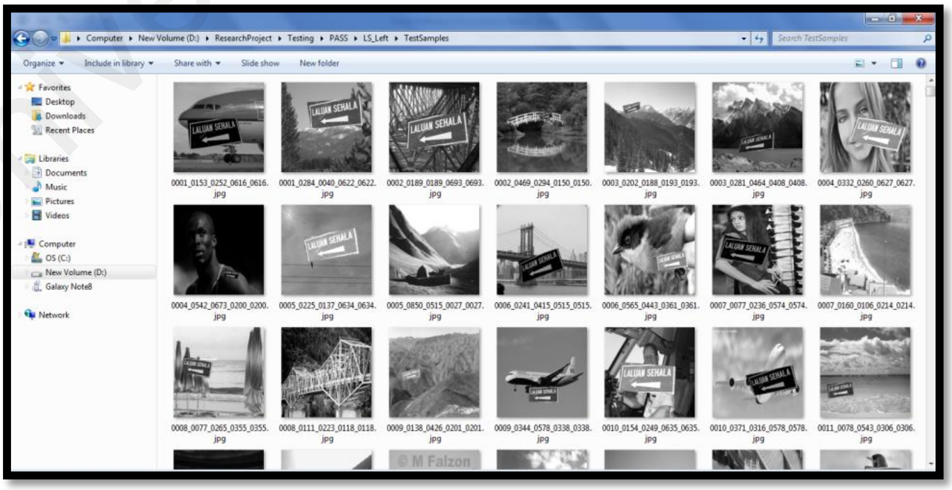
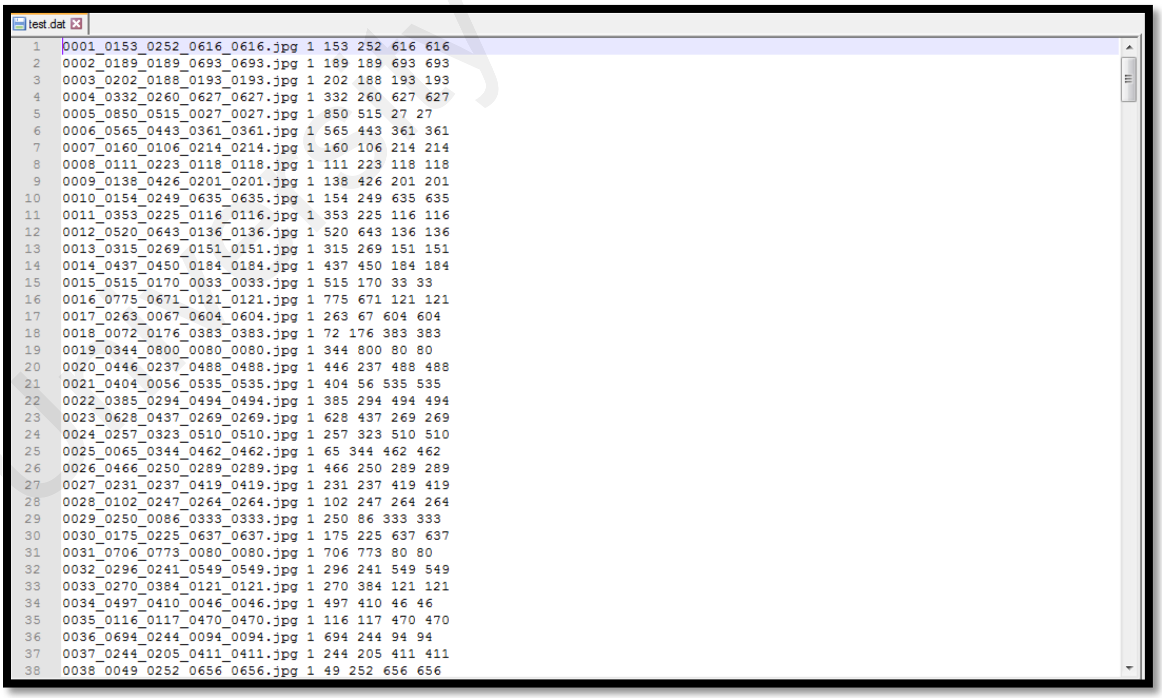


Figure 3.15: Created test samples

Figure 3.16: An example of created sample


test.dat

```
0001_0153_0252_0616_0616.jpg 1 153 252 616 616
0002_0189_0189_0693_0693.jpg 1 189 189 693 693
0003_0202_0188_0193_0193.jpg 1 202 188 193 193
0004_0332_0260_0627_0627.jpg 1 332 260 627 627
0005_0850_0515_0027_0027.jpg 1 850 515 27 27
0006_0565_0443_0361_0361.jpg 1 565 443 361 361
0007_0160_0106_0214_0214.jpg 1 160 106 214 214
0008_0111_0223_0118_0118.jpg 1 111 223 118 118
0009_0138_0426_0201_0201.jpg 1 138 426 201 201
0010_0154_0249_0635_0635.jpg 1 154 249 635 635
0011_0353_0225_0116_0116.jpg 1 353 225 116 116
0012_0520_0643_0136_0136.jpg 1 520 643 136 136
0013_0315_0269_0151_0151.jpg 1 315 269 151 151
0014_0437_0450_0184_0184.jpg 1 437 450 184 184
0015_0515_0170_0033_0033.jpg 1 515 170 33 33
0016_0775_0671_0121_0121.jpg 1 775 671 121 121
0017_0263_0067_0604_0604.jpg 1 263 67 604 604
0018_0072_0176_0383_0383.jpg 1 72 176 383 383
0019_0344_0800_0080_0080.jpg 1 344 800 80 80
0020_0446_0237_0488_0488.jpg 1 446 237 488 488
0021_0404_0056_0535_0535.jpg 1 404 56 535 535
0022_0385_0294_0494_0494.jpg 1 385 294 494 494
0023_0628_0437_0269_0269.jpg 1 628 437 269 269
0024_0257_0323_0510_0510.jpg 1 257 323 510 510
0025_0065_0344_0462_0462.jpg 1 65 344 462 462
0026_0466_0250_0289_0289.jpg 1 466 250 289 289
0027_0231_0237_0419_0419.jpg 1 231 237 419 419
0028_0102_0247_0264_0264.jpg 1 102 247 264 264
0029_0250_0086_0333_0333.jpg 1 250 86 333 333
0030_0175_0225_0637_0637.jpg 1 175 225 637 637
0031_0706_0773_0080_0080.jpg 1 706 773 80 80
0032_0296_0241_0549_0549.jpg 1 296 241 549 549
0033_0270_0384_0121_0121.jpg 1 270 384 121 121
0034_0497_0410_0046_0046.jpg 1 497 410 46 46
0035_0116_0117_0470_0470.jpg 1 116 117 470 470
0036_0694_0244_0094_0094.jpg 1 694 244 94 94
0037_0244_0205_0411_0411.jpg 1 244 205 411 411
0038_0049_0252_0656_0656.jpg 1 49 252 656 656
```

Figure 3.17: An example of .dat file

3.7.2 Image Testing

After the test samples have been created, another batch file to call 'opencv_performance.exe' is composed. This batch file is to run the utility in command prompt and execute the testing based on the testing samples which have been created. Refer to figure below, note that the test samples created are saved in the 'TestSamples' folder. The trained classifier is now in .xml file. While the batch file in .bat format is to execute the OpenCV utility, 'opencv_performance.exe'. The work folder of testing and command in batch file are illustrated in figures below.

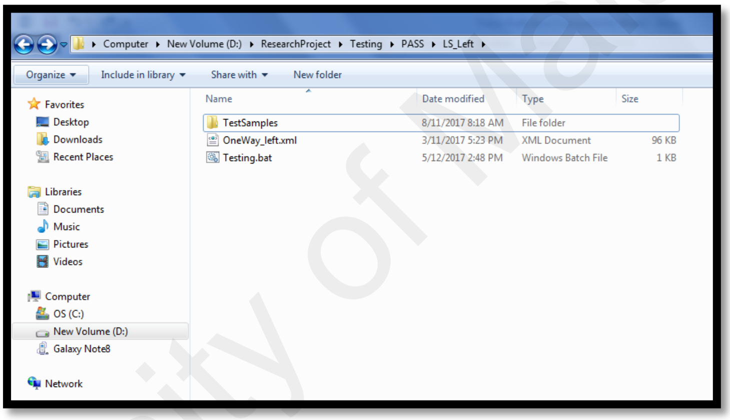


Figure 3.18: Work folder for testing

```
C:\opencv\build\x64\vc12\bin\opencv_performance.exe -data OneWay_left.xml -info
TestSamples\test.dat -sf 1.2 -w 15 -h 20 > TestResult.log
```

Figure 3.19: Testing batch file

From the command shown in figure 3.19, a log file with the name of 'TestResult' will be generated once the testing is completed, as shown in figure below.

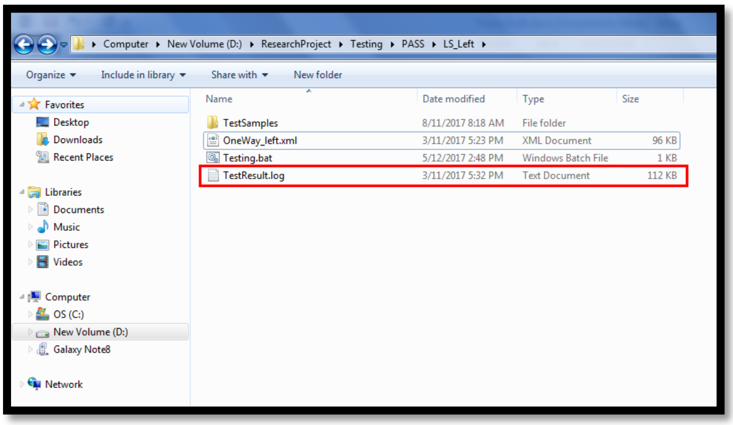


Figure 3.20: Newly generated log file for testing results

Besides, the output images will also be generated. These images have the rectangular marking. These markings indicate that the classifier have identified the positive objects appeared on the image, as shown below.

Figure 3.21: Testing results

The details of the results will be discussed in the following chapter.

3.8 Python Script

Python script is needed in order to run the trained classifier in real life detection or also known as video testing. All .xml files corresponding to classifier of each sign will be loaded into the script. The python script will use these .xml files as library and run the detection through the live feed from webcam. The complete script is attached in appendix.

The reason of running detection through the live feed from webcam is to analyze the performance of the trained classifiers in real life. To carry out the testing, the webcam is mounted on the dashboard of the car and it is connected to the laptop. Then the testing was conducted while the car is moving. Testing for each classifiers were conducted about 10 times, in day and night time respectively.

CHAPTER 4: RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter presents the results obtained. On top of that, the analysis and the discussion of the results are covered in this chapter too.

4.2 Results

The results can be divided into two sections. Section 1 presents and discusses about the testing results generated by the 'opencv_performance.exe'. Section 2 presents and discusses about on-the-road testing results.

4.2.1 Image Testing Results

In this section, the results generated by the 'opencv_performance.exe' are presented. As mentioned in previous chapter, results log will be generated once the testing is done. Figures below show the result logs generated for classifier of one way (left) road sign.

File Name	Hits	Missed	False
0001_0064_0204_0537_0537.jpg	1	0	8
0002_0330_0149_0311_0311.jpg	1	0	9
0003_0155_0286_0601_0601.jpg	1	0	3
0004_0351_0578_0342_0342.jpg	1	0	13
0005_0121_0230_0383_0383.jpg	1	0	4
0006_0697_0281_0191_0191.jpg	1	0	7
0007_0348_0532_0335_0335.jpg	1	0	12
0008_0611_0296_0223_0223.jpg	1	0	10
0009_0189_0332_0570_0570.jpg	1	0	3
0010_0722_0596_0061_0061.jpg	0	1	10
0011_0177_0291_0416_0416.jpg	1	0	42
0012_0107_0162_0548_0548.jpg	1	0	4
0013_0593_0156_0082_0082.jpg	1	0	13
0014_0444_0120_0386_0386.jpg	1	0	17
0015_0470_0126_0379_0379.jpg	1	0	9
0016_0489_0617_0294_0294.jpg	1	0	10
0017_0083_0392_0504_0504.jpg	1	0	4
0018_0247_0207_0493_0493.jpg	1	0	2
0019_0236_0278_0627_0627.jpg	1	0	1
0020_0388_0139_0299_0299.jpg	1	0	13

Figure 4.1: Upper part Results Log for One Way (left) Classifier

```
+---------------------------------+-------+------+------+
|     0983_0092_0279_0617_0617.jpg|      1|     0|     5|
+---------------------------------+-------+------+------+
|     0984_0266_0281_0643_0643.jpg|      1|     0|     4|
+---------------------------------+-------+------+------+
|     0985_0260_0445_0274_0274.jpg|      1|     0|     2|
+---------------------------------+-------+------+------+
|     0986_0493_0143_0140_0140.jpg|      1|     0|    23|
+---------------------------------+-------+------+------+
|     0987_0298_0044_0589_0589.jpg|      1|     0|     5|
+---------------------------------+-------+------+------+
|     0988_0369_0101_0280_0280.jpg|      1|     0|    40|
+---------------------------------+-------+------+------+
|     0989_0496_0399_0245_0245.jpg|      1|     0|    13|
+---------------------------------+-------+------+------+
|     0990_0333_0391_0076_0076.jpg|      1|     0|    35|
+---------------------------------+-------+------+------+
|     0991_0205_0401_0212_0212.jpg|      1|     0|    12|
+---------------------------------+-------+------+------+
|     0992_0456_0253_0370_0370.jpg|      1|     0|     5|
+---------------------------------+-------+------+------+
|     0993_0067_0251_0655_0655.jpg|      1|     0|     6|
+---------------------------------+-------+------+------+
|     0994_0206_0268_0112_0112.jpg|      1|     0|     1|
+---------------------------------+-------+------+------+
|     0995_0122_0266_0636_0636.jpg|      1|     0|     2|
+---------------------------------+-------+------+------+
|     0996_0369_0732_0158_0158.jpg|      1|     0|     8|
+---------------------------------+-------+------+------+
|     0997_0204_0255_0609_0609.jpg|      1|     0|     1|
+---------------------------------+-------+------+------+
|     0998_0506_0440_0396_0396.jpg|      1|     0|     6|
+---------------------------------+-------+------+------+
|     0999_0168_0469_0256_0256.jpg|      1|     0|    20|
+---------------------------------+-------+------+------+
|     1000_0512_0179_0336_0336.jpg|      1|     0|     1|
+---------------------------------+-------+------+------+
|                            Total|    955|    45| 11939|
+=================================+=======+======+======+
Number of stages: 7
Number of weak classifiers: 12
Total time: 127.561000
```

Figure 4.2: Last Part of Results Log for One Way (left) Classifier

Table below summarizes the results for all classifiers.

Table 4.1: Results for Classifier Testing by 'opencv_performance.exe'

Classifier	Total number of test samples	Hits	Missed	False	Accuracy (%)
Stop	1000	998	2	922	99.8
One way (left)	1000	952	48	1043	95.2
One way (right)	1000	944	56	2812	94.4
Bump ahead	1000	998	2	4336	99.8
Traffic light ahead	1000	919	81	3798	91.9
No Overtaking	1000	978	22	4178	97.8
No Stopping	1000	981	19	6918	98.1
No U-turn	2000	1934	66	579	96.7
No Parking	1000	998	2	300	99.8
No Entry	1000	952	48	1043	95.2
No Left Turn	2000	1960	40	895	98.0
60km/h speed limit	1000	998	2	802	99.8

From the table above and figure 4.1, it is noticeable that there are hits, missed and false. The 'hit' means that the classifier has successfully located the object to be detected in the image. Each test sample carries only one particular sign in the image. Thereby, if the object can be detected by the classifier, the hit in that particular image will be mark as 1, or else it will be categorized as missed. So, the accuracy is calculated as follows:

$$\text{Accuracy} = \frac{Hits}{Number\ of\ testing\ samples} \times 100\%$$

One of the output images from the testing for one way (left) classifier is shown in figure below.

Figure 4.3: Successful detection

Figure above shows that the testing result of one way (left) classifier on this image is hit = 1, missed = 0 and false = 0. It means in this testing, the classifier has successfully detect the object correctly. Another example where the classifier has missed the detection of the sign is illustrated in figure below.

Figure 4.4: Missed detection

Figure 4.4 is an example of missed detection. In this image, the classifier could not detect the presence of the sign. So in this case, this picture will have 1 in missed column and 0 in hits column. Figure below shows another example of false detection.

Figure 4.5: False detection

In figure 4.5 above, it can be seen that the classifier can detect the correct sign but at the same time, there were two other detections. The two detections were false positive detections because there are no signs in the highlighted region. While the middle highlighted region is correction detection. In this case, the hit is 1 while the other two detections belong in false column.

4.2.2 Video Testing Results

The testing results are summarized in the table below. Each testing were carried out in a way that the webcam is mounted on dashboard of the car and the script is run when the car is moving. The script has compiled all the 12 classifiers and when it runs, it runs all the 12 classifiers concurrently. Car has been driven on some specific routes.

These routes contain the signs to be detected. Classifiers for each signs have been tested averagely 20 times, 10 times in day time and 10 times at night. The accuracy of the classifiers are calculated by calculating how many times the classifier can detect the traffic signs on the road, dividing by how many times they are tested. Results are summarized in table below.

Table 4.2: Accuracy of classifier for on the video feed testing

Classifier	Successful detection		Accuracy (%)
	Night	Morning	
Stop	9	7	80
One way (left)	8	8	80
One way (right)	9	8	85
Bump ahead	10	9	95
Traffic light ahead	9	9	85
No Overtaking	9	7	80
No Stopping	9	8	85
No U-turn	9	8	85
No Parking	10	7	85
No Entry	9	8	85
No Left Turn	9	7	80
60km/h speed limit	9	7	80
		Average Accuracy	83.33

Examples of successful detections are illustrated in the figures below:

Figure 4.6: Successful detection of 60km/h speed limit sign at night

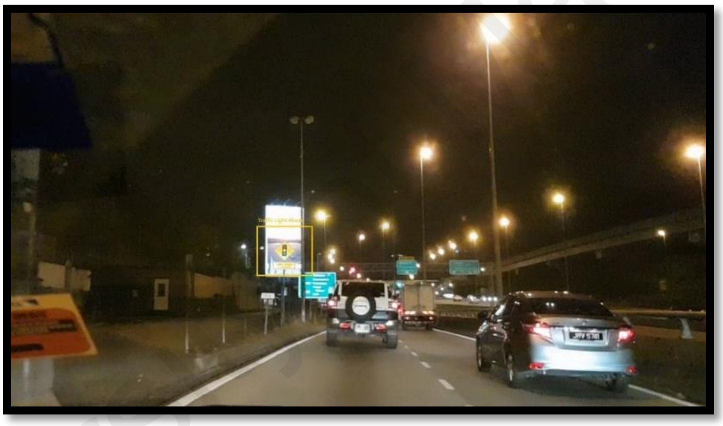


Figure 4.7: Successful detection of traffic light sign at night

Figure 4.8: Successful detection of traffic light sign in day time

Figure 4.9: Successful detection of one way (right) in day time

Figure 4.10: Successful detection of bump sign in day time

Figure 4.11: Successful detection of No U-turn sign in day time

Figure 4.12: Successful detection of No U-turn sign in rainy day

Figure 4.13: Successful detection of One Way (Left)

Figure 4.14: Successful detection of No Entry sign

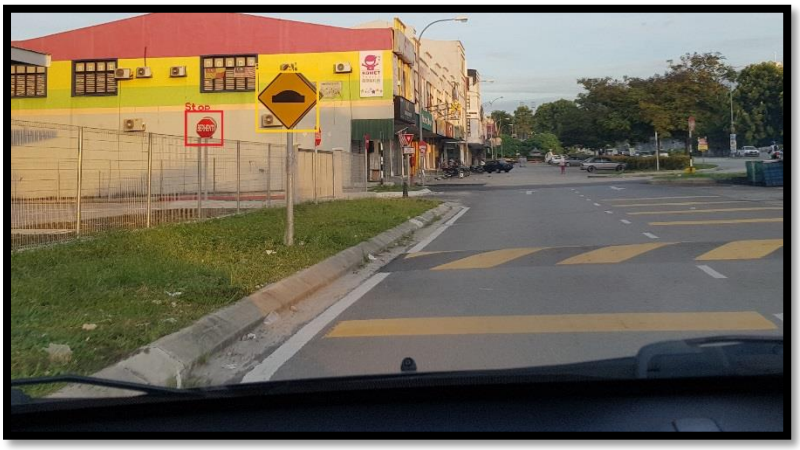


Figure 4.15: Successful detection of Bump Ahead and Stop sign

Figure 4.16: Successful detection of Bump Ahead at Night

Figure 4.17: Successful detection of two traffic light ahead signs

Figure 4.18: Successful detection of no stopping sign

Figure 4.19: Successful detection 60km/h speed limit sign and no stopping sign

Figure 4.20: Successful detection no parking sign

Misclassification or wrong detection happened as well. However, most of the wrong detection happened at night time. Figures below showed the example of wrong detection.

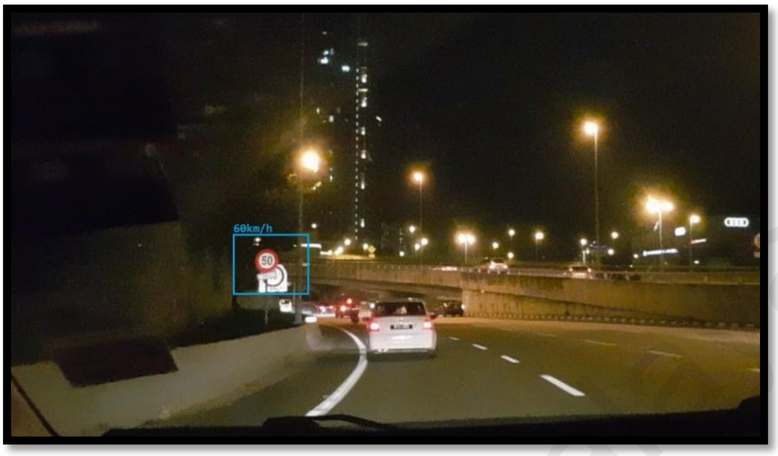


Figure 4.21: Wrong detection at night time part 1

Figure 4.22: Wrong detection at night time part 2

Figures above show the system mistakenly classified 50km/h speed limit sign as 60km/h. The system could only detect it was not the correct sign when the car moved closer to the sign.

Figure 4.23: Miss Detection of Stop Sign

Figure above shows that the system was unable to detect the stop sign. This is because part of the sign was covered by the tree leaves. That resulted in failure of detection.

Figure 4.24: False detection and wrong detection

False positive detection happened in few times during testing at night. The example can be seen from figure above. Figure above shows wrong detection, which is the stop sign. The system has wrongly detected the sign to be a stop sign. While the system has also detected a bump sign in the image which is not existing. Thereby, it is classified as false positive detection.

4.3 Performance Analysis

From the results obtained in the testing carried out by 'opencv_performace.exe' utility, the classifiers have shown average accuracy higher than 90%. The trained classifiers have also achieved average accuracy of 83.33% when they were tested to detect the traffic signs on the road via webcam feed. It is noticeable that in day time, the accuracy of all classifiers are averagely higher. It is believed that because the variation in illumination is relatively smaller compared to night time. At night, the complexity of the environment and illumination becomes much higher due to different light sources on the street. Besides of the illumination condition, the classifiers encounter problem in recognizing the signs which have vandalized and covered by tree leaves and advertisement poster. The vandalism have caused the outlook of the signs to be very badly altered and the advertisement posters cover some of the important features that can be recognized by computer.

CHAPTER 5: CONCLUSION

5.1 Introduction

In this chapter, research conclusions and future works recommendations are discussed. Initially, research summaries in relations to the research objectives are given followed by research conclusions. Finally, several recommendations for further research works are presented.

5.2 Research Conclusions

Cascade training for traffic signs detection based on Haar-like features have been proposed in this study. The reason cascaded training based on Haar-like features is proposed is because of shorter training time. Furthermore, it is also proven that classifier based on Haar-features is sufficient to create classifiers for traffic signs with high accuracy based on the testing conducted.

In relation to the study objectives:

1. This research has successfully developed traffic signs detection system based on Haar-like features cascade classifier.
2. The performance and accuracy of proposed system has been studied and analysed.

5.3 Recommendations for Future Works

As mentioned in previous chapters, the performance of trained classifier will be affected by environment factors such as illumination condition. Development of the classifier that can work independently without being affected by external environment factors should be considered in the future work. This is because the safety will be greatly affected by the robustness of the system.

Furthermore, there is still a big room of improvement for the accuracy of the detection system. One of the suggestion to improve the accuracy is to increase the number of dataset. However, this will also increase training time. Other training methods or statistical models should be considered too.

Lastly, the detection and recognition system is recommended and suggested to have self-learning and self-testing algorithm. This suggestion is made because the signs in every country vary with each other. Thereby, self-learning in classifying sign with different outlook but carrying same message should is essential.

REFERENCES

[1] Viola, P., & Jones, M. (n.d.). Robust real-time face detection. Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001. doi:10.1109/iccv.2001.937709

[2] Verschae, R., & Ruiz-Del-Solar, J. (2015). Object Detection: Current and Future Directions. Frontiers in Robotics and AI, 2. doi:10.3389/frobt.2015.00029

[3] M. A. Fischler and R. A. Elschlager, "The Representation and Matching of Pictorial Structures," in *IEEE Transactions on Computers*, vol. C-22, no. 1, pp. 67-92, Jan. 1973. doi: 10.1109/T-C.1973.223602

[4] E. Osuna, R. Freund and F. Girosit, "Training support vector machines: an application to face detection," *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Juan, 1997, pp. 130-136.
doi: 10.1109/CVPR.1997.609310

[5] H. A. Rowley, S. Baluja and T. Kanade, "Neural network-based face detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23-38, Jan 1998. : 10.1109/34.655647

[6] K. K. Sung and T. Poggio, "Example-based learning for view-based human face detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 39-51, Jan 1998. doi: 10.1109/34.655648

[7] H. Schneiderman and T. Kanade, "A statistical method for 3D object detection applied to faces and cars," *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, Hilton Head Island, SC, 2000, pp. 746-751 vol.1. doi: 10.1109/CVPR.2000.855895

[8] H. Sahbi, D. Geman and N. Boujemaa, "Face detection using coarse-to-fine support vector classifiers," *Proceedings. International Conference on Image Processing*, 2002, pp. 925-928 vol.3. doi: 10.1109/ICIP.2002.1039124

[9] S. Romdhani, P. Torr, B. Scholkopf and A. Blake, "Computationally efficient face detection," *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, Vancouver, BC, 2001, pp. 695-700 vol.2. doi: 10.1109/ICCV.2001.937694

[10] S. Z. Li and Zhenqiu Zhang, "FloatBoost learning and statistical face detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1112-1123, Sept. 2004. doi: 10.1109/TPAMI.2004.68

[11] Freund, Y. and Schapire, R. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1), pp.119-139.

[12] Chang Huang, Haizhou Ai, Yuan Li and Shihong Lao, "Vector boosting for rotation invariant multi-view face detection," *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, 2005, pp. 446-453 Vol. 1. doi: 10.1109/ICCV.2005.246

[13] Verschae, R., Ruiz-Del-Solar, J., & Correa, M. (2007). A unified learning framework for object detection and classification using nested cascades of boosted classifiers. Machine Vision and Applications, 19(2), 85-103. doi:10.1007/s00138-007-0084-0

[14] Mutch, J., & Lowe, D. G. (2008). Object Class Recognition and Localization Using Sparse Features with Limited Receptive Fields. International Journal of Computer Vision, 80(1), 45-57. doi:10.1007/s11263-007-0118-0

[15] R. Sivalingam, G. Somasundaram, V. Morellas, N. Papanikolopoulos, O. Lotfallah, and Y. Park, “Dictionary learning based object detection and counting in traffic scenes,” *Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras - ICDSC 10*, 2010.

[16] A. Jain, L. Zappella, P. Mcclure, and R. Vidal, “Visual Dictionary Learning for Joint Object Categorization and Segmentation,” *Computer Vision – ECCV 2012 Lecture Notes in Computer Science*, pp. 718–731, 2012.

[17] C. H. Lampert, M. B. Blaschko and T. Hofmann, "Efficient Subwindow Search: A Branch and Bound Framework for Object Localization," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2129-2142, Dec. 2009. doi: 10.1109/TPAMI.2009.144

[18] P. F. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627-1645, Sept. 2010. doi: 10.1109/TPAMI.2009.167

[19] Buduma, N. (2016). Fundamentals of Deep Learning: Designing Next-Generation Artificial Intelligence Algorithms. Sebastopol: OReilly Media.

[20] Deng, L., & Yu, D. (2014). Deep Learning: Methods and Applications. Foundations and Trends® in Signal Processing, 7(3-4), 197-387. doi:10.1561/2000000039

[21] Goodfellow, I., Bengio, Y., & Courville, A. (2017). Deep learning. Cambridge, MA: The MIT Press.

[22] W. Ouyang and X. Wang, "Joint Deep Learning for Pedestrian Detection," *2013 IEEE International Conference on Computer Vision*, Sydney, VIC, 2013, pp. 2056-2063. doi: 10.1109/ICCV.2013.257

[23] Leitner, J., Harding, S., Chandrashekhariah, P., Frank, M., Förster, A., Triesch, J., & Schmidhuber, J. (2013). Learning visual object detection and localisation using icVision. Biologically Inspired Cognitive Architectures, 5, 29-41. doi:10.1016/j.bica.2013.05.009

[24] J. H. Shi and H. Y. Lin, "A vision system for traffic sign detection and recognition," *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, Edinburgh, 2017, pp. 1596-1601. doi: 10.1109/ISIE.2017.8001485

[25] S. J. Zabihi, S. M. Zabihi, S. S. Beauchemin and M. A. Bauer, "Detection and recognition of traffic signs inside the attentional visual field of drivers," *2017 IEEE Intelligent Vehicles Symposium (IV)*, Los Angeles, CA, 2017, pp. 583-588. doi: 10.1109/IVS.2017.7995781

[26] Wahyono, L. Kurnianggoro, J. Hariyono and K. H. Jo, "Traffic sign recognition system for autonomous vehicle using cascade SVM classifier," *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, Dallas, TX, 2014, pp. 4081-4086. doi: 10.1109/IECON.2014.7049114

[27] M. Karaduman and H. Eren, "Deep learning based traffic direction sign detection and determining driving style," *2017 International Conference on Computer Science and Engineering (UBMK)*, Antalya, 2017, pp. 1046-1050. doi: 10.1109/UBMK.2017.8093453

[28] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler, "A system for traffic sign detection, tracking, and recognition using color, shape, and motion information," In *Intelligent Vehicles Symposium*, pp. 255-260, 2005.

[29] I. M. Creusen, R. G. Wijnhoven, E. Herbschleb, and P. H. N. de With, "Color exploitation in hog-based traffic sign detection," In *Image Processing (ICIP)*, pp. 2669-2672, 2010.

APPENDIX A

Python Script

```
import numpy as np
import cv2

# No U-TURN Cascade
NoUturn_cascade_1 = cv2.CascadeClassifier('NoUturn_1.xml')

# No STOPPING Cascade
NoStopping_cascade_1 = cv2.CascadeClassifier('NoStopping_1.xml')

# STOP Cascade
Stop_cascade_2 = cv2.CascadeClassifier('Stop_2.xml')

# No LEFT TURN Cascade
NoLeftTurn_cascade_1 = cv2.CascadeClassifier('NoLeftTurn_1.xml')

# 60km/h SPEED LIMIT Cascade
_60kmph_cascade_1 = cv2.CascadeClassifier('60km_1.xml')

# Laluan Sehala Cascade
LS_Right_cascade_1 = cv2.CascadeClassifier('LS_Right.xml')

# Laluan Sehala Cascade
LS_Left_cascade_1 = cv2.CascadeClassifier('LS_Left_1.xml')

# Bump Ahead Cascade
Bump_Ahead_1 = cv2.CascadeClassifier('Bump_1.xml')

# Traffic Light Ahead Cascade
Traffic_light_Ahead_1 = cv2.CascadeClassifier('TL_1.xml')

# No overtaking Cascade
No_Overtaking_1 = cv2.CascadeClassifier('NO_1.xml')

# No Parking Cascade
No_Parking = cv2.CascadeClassifier('NP_2.xml')

# No Entry Cascade
No_Entry = cv2.CascadeClassifier('NE_1.xml')
```

```
cap = cv2.VideoCapture(0)

while 1:
	ret, img = cap.read()
	gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

	 # image, reject levels level weights.
	 NoUturn_1 = NoUturn_cascade_1.detectMultiScale(gray, 2.5, 5)

	NoStopping_1 = NoStopping_cascade_1.detectMultiScale(gray, 2.5, 5)

	Stop_2 = Stop_cascade_2.detectMultiScale(gray, 2.2, 5)

	NoLeftTurn_1 = NoLeftTurn_cascade_1.detectMultiScale(gray, 1.78, 5) #Need retrain

	_60kmph_1 = _60kmph_cascade_1.detectMultiScale(gray, 2.5, 5)

	LS_Right = LS_Right_cascade_1.detectMultiScale(gray, 1.5, 5)

	LS_Left = LS_Left_cascade_1.detectMultiScale(gray, 1.3, 5)

	Bump_Ahead = Bump_Ahead_1.detectMultiScale(gray, 2.2, 5)

	Traffic_Light_Ahead = Traffic_light_Ahead_1.detectMultiScale(gray, 1.43, 5)

	No_Overtaking = No_Overtaking_1.detectMultiScale(gray, 1.9, 5)

	No_Parking_1 = No_Parking.detectMultiScale(gray, 1.9, 5)

	No_Entry_1 = No_Entry.detectMultiScale(gray, 1.9, 5)

	# add this
	for (x,y,w,h) in NoUturn_1:
		cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)
		font = cv2.FONT_HERSHEY_SIMPLEX
		cv2.putText(img,'No U-TURN',(x,y), font, 0.5, (0,0,255), 1, cv2.LINE_AA)

	# For No Stopping Road Sign
	for (x,y,w,h) in NoStopping_1:
```

```
            cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img,'No Entry',(x,y), font, 0.5, (0,0,255), 1, cv2.LINE_AA)

        # For Stop Road Sign
        for (x,y,w,h) in Stop_2:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)
            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img,'STOP',(x,y), font, 0.5, (0,0,255), 1, cv2.LINE_AA)

        # For No left turn road sign
        for (x,y,w,h) in NoLeftTurn_1:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,255),2)
            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img,'No LEFT TURN',(x,y), font, 0.5, (255,0,255), 1,
cv2.LINE_AA)

        # For One Way Right road sign
        for (x,y,w,h) in LS_Right:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,255),2)
            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img,'One Way Left',(x,y), font, 0.5, (0,0,255), 1, cv2.LINE_AA)

        # For One Way Left road sign
        for (x,y,w,h) in LS_Left:
            cv2.rectangle(img,(x,y),(x+w,y+h),(128,128,128),2)
            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img,'One Way Left',(x,y), font, 0.5, (0,0,255), 1, cv2.LINE_AA)

        # For 60km/h speed limit road sign
        for (x,y,w,h) in _60kmph_1:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)
            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img,'60km/h',(x,y), font, 0.5, (255,255,0), 1, cv2.LINE_AA)
```

```
        # For Bump Ahead road sign
        for (x,y,w,h) in Bump_Ahead:
            cv2.rectangle(img,(x,y),(x+w,y+h),(0,140,225),2)
            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img,'Bump Ahead',(x,y), font, 0.5, (0,140,255), 1, cv2.LINE_AA)

        # For Traffic Light Ahead road sign
        for (x,y,w,h) in Traffic_Light_Ahead:
            cv2.rectangle(img,(x,y),(x+w,y+h),(0,128,225),2)
            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img,'Traffic Light Ahead',(x,y), font, 0.5, (0,128,255), 1, cv2.LINE_AA)

        # For No Overtaking road sign
        for (x,y,w,h) in No_Overtaking:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,100,225),2)
            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img,'Overtaking not allowed',(x,y), font, 0.5, (255,100,255), 1, cv2.LINE_AA)

        # For No Parking road sign
        for (x,y,w,h) in No_Parking_1:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,100,225),2)
            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img,'No Parking',(x,y), font, 0.5, (1,1,255), 1, cv2.LINE_AA)

        # For No Entry Sign
        for (x,y,w,h) in No_Entry_1:
            cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img,'No Entry',(x,y), font, 0.5, (255,0,0), 1, cv2.LINE_AA)


        cv2.imshow('img',img)
        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break

cap.release()
cv2.destroyAllWindows()
```