New Explorations in Electrical Engineering (NECE)
Department of Electrical Engineering,
Faculty of Engineering
Universiti Malaya, MALAYSIA

Vol. 1, No. 2 (2025)
pp. 11–33

Real-Time Text Extraction from Malaysian Road Direction Signs Using Dashboard Camera

Lim Wan Chin, Noraisyah Mohamed Shah* and Wan Amirul Wan Mohd Mahiyiddin
Department of Electrical Engineering, Faculty of Engineering, Universiti Malaya, 50603 Kuala Lumpur, Malaysia
E-mail: noraisyah@um.edu.my

Abstract. The road direction sign, which serves as a navigation tool, aids the driver in travelling on the road. A real-time text extraction from road direction signs is investigated, using input video images acquired from a dashboard camera while driving on Malaysian roads. This paper proposed a three-stage approach for real-time text extraction: (i) traffic direction signboard detection; (ii) text detection; and (iii) text recognition. In the direction signboard detection, the YOLOv5s model was applied and achieved a precision of 97%and a recall of 96%. In the text detection stage, three text detection models, EAST, DB and PAN, were investigated to select the best text detection model. The DB model is chosen with the highest IoU of 0.8238 and the shortest inference time of 0.1938s. In the text recognition stage, four models (Pytesseract, CRNN, RobustScanner and ABINet) are employed to perform evaluation and pick the best model. The CRNN is selected with the lowest character error rate of 5.85%, the highest accuracy of 97.84%, and a fast inference time of 0.324s. The proposed models, YOLOv5s, DB, and CRNN are combined to run sequentially in the real-time system during daytime conditions. It is performed by using a laptop that is equipped with a GPU and programmed in Python. The real-time text extraction system can run at an adaptive frame rate of 6.5 frames per second with an accuracy of 97%.

Keywords. Road Sign Detection, Text Detection, Text Recognition, Real-time Implementation.

1. Introduction

An autonomous car is a vehicle that utilises modern vehicle safety technologies such as computers, cameras, and sensors to drive itself without human control. It can sense the environment and perform tasks such as tracking other vehicles, looking for pedestrians, lane detection and traffic sign detection. The road direction sign contains words, texts, arrow signs and numbers which provide information to the drivers as they navigate their way. However, autonomous cars are not equipped with road sign detection and text extraction (Swetha and Sivakumar 2021). Therefore, with this additional feature, the autonomous car can acquire additional information from the sensors or camera to provide a better understanding of the current environment and generate appropriate predictions for the path navigation. By obtaining sufficient information and extracting relevant information from the surroundings, a more accurate and reliable driving system for autonomous cars is obtained (Iqbal 2020).

Vision plays an important role in driving where it helps the driver interact with the surrounding environment. The head of Programme Development at Mission for Vision stated that “During 2019-2020, they screened 15,000 drivers across 12 states; 40% of them were found to have uncorrected refractive errors and drivers with poor eyesight had an 81% road crash involvement rate.” As a result, for those visually impaired drivers, disabled drivers, or elderly drivers may need a human driving assistance

technology to assist them to obtain the real-time safety information during times of uncertainty (Rolison et al. 2018).

Real-time text extraction systems from road signs is one of the applications of human driving assistance technology. It helps those drivers to retrace the road sign that they missed and provides a better visualization of the road sign at night (Agrawal 2017). Therefore, human assistance driving technology plays an important role in guiding drivers to remain focused on the road, resulting in better driver safety and reducing the accident rate involving those visually impaired or elderly drivers.

Typically, road signs are installed in an outdoor environment. Due to the influence of ultraviolet rays and rain, the text on the road sign will fade, making it difficult to read. Currently, a health check of the road sign is operated manually by the transportation authorities. It can be automated by using the real-time implementation of a text extraction system on the road signs. Moreover, this system can be used as a data collection platform to check whether the road signs is obstructed by any obstacle, or if any defection happens on it. Thus, the authorities in charge can easily track the condition of the road sign (Chincholkar and Kumar 2019).

The Vienna Convention on Road Signs and Signals defines the "Informative Signs" class as signs that are intended to guide road-users while they are travelling or to provide them with other information which may be useful, such as giving distances or directions to a given location, as well as information, facilities, or service signs. These signs vary from one country to another as the Vienna Convention does not specify sizes, colours, symbols or positions of such sign. However, direction signs must be either a rectangle or an arrow shaped pentagon, and they may not contain placenames in more than two languages. In Malaysia, the direction sign uses the English alphabets, the blue background is used for federal roads, state roads, and municipal roads, while the green background is used for toll expressways or highways. They are located at the sides of the road as well as overhead. For brevity, these road direction signs are referred to as road signs.

To implement a real-time text extraction from the road direction sign captured using dashboard cameras (dashcam), different models for road sign detection, text detection and text extraction are investigated, and based on their performance the best combination of models are chosen for the real-time implementation.

2. Related Works

In this section, recent developments in the field of object detection, text detection and text recognition are presented.

2.1. Object Detection

Object detection is a computer vision technique that allows the system to estimate the position of an object based on an image or video. The frameworks of object detection techniques can be divided into two main categories, which are algorithms-based classification or algorithms-based regression. CNN and RNN are classified as algorithms-based classifications by identifying the area of interest in the image and performing classification using algorithms. This approach is not recommended since it takes a long time to predict each detectable region. For algorithms-based regression, it will first allocate the target of interest from the image, then determine the category of the target and form a bounding box. Thus, the process of detection will be faster as compared to classification algorithms (Geethapriya et al. 2019). This shows that algorithm-based classification, such as You Only Look Once (YOLO) works well as compared to algorithm-based regression.

According to (Wang et al. 2021), YOLO is a robust real-time object detection system that can detect various classes within a short period of time. YOLO only owns a single process by splitting the images into regions and predicting the bounding boxes using a single neural network. Unlike the R-CNN model, which requires thousands of network evaluations for each image, the YOLO model simply requires a single network assessment for each image. With this unique structure, YOLO can perform 1000 times faster than R-CNN and 100 times faster than Fast R-CNN. According to the research by (Nepal and Eslamiat 2022), a comparison of YOLOv3 and Single Shot Detector (SSD) was performed. YOLOv3 has a higher F1-score and FPS as compared to the SSD. Moreover, the YOLOv3 can achieve good detection speed and accuracy on real-time applications.

Based on (Redmon et al. 2016), YOLO is still able to achieve the highest mean average precision (mAP) for real-time object detection. The Deformable parts model (DPM) which is one of the real-time detectors, employs a sliding window method to conduct detection. The performance of both detectors has been compared. The DPM detector has a mAP that is two times lower than YOLO. Although the DPM detector can perform in real-time, this clearly demonstrates its poor performance in terms of accuracy. Similarly, (Khalid et al. 2024) proposed a multi-stage framework for roadside sign text extraction. Their system leverages a customized YOLOv5s detector for highly accurate signboard localization, outperforming alternative state-of-the-arts methods. For subsequent text localization, they utilize preprocessing techniques paired with Maximally Stable Extremal Regions (MSER) to ensure robust performance under degraded visibility. In short, YOLO can adapt easily to a new environment, making it ideal for applications that require fast and robust object detection.

2.2. Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is a technology for converting visual text into machine-readable text that may be edited. These images can be in handwritten text, printed text, and the natural scene photograph (Sabu and Das 2018). OCR can be divided into two main parts which are text detection and text recognition. Text detection aims to localise the bounding box of the text within the input image. Text localization is crucial to perform the second part of OCR which is text recognition, in which the text is extracted from the input image. OCR can be performed using either conventional computer vision techniques or more advanced deep learning techniques. There are many open-source tools available to perform OCR, for example Tesseract-OCR, PaddleOCR, MMOCR and TrOCR. The accuracy rate of any OCR open-source tool varies from 71% to 98% (Patel et al. 2012).

2.2.1. Text Recognition

The main function of text detection is to distinguish text from backgrounds with various conditions of typeface, shape, illumination, orientation, or being blurred (Kulkarni and Barbadekar 2017). One of the challenges for text detection is a realistic scene with different fonts, multiple colors, and arbitrary shapes. Besides, different cameras capture images with various orientations, resolutions, contrast rations, illuminations, and noises for the same text scene. All these unknown factors may make some of the algorithms fail to detect the text from the input image (Wang et al. 2021).

2.2.1.1. Efficient and Accuracy Scene Text detection (EAST)

In (Zhou et al. 2017), a new detection method named Efficient and Accuracy Scene Text detection (EAST) is proposed, which consists of two stages: a Fully Convolutional Network (FCN) and a Non-Maximum Suppression (NMS) merging stage. The EAST detection pipeline employs an FCN model that generates word or text-line level predictions directly, by eliminating the need for redundant and slow intermediate steps. The EAST model achieves end-to-end text region detection, which greatly improves text detection accuracy and speed (Lu et al. 2021). Qualitative and quantitative experiments are conducted

by using three public benchmarks: ICDAR2015, COCO-Text and MSRA-TD500. The EAST model achieves an F-score of 78.20% on ICDAR 2015, 76.08% on MSRA-TD500 and 39.45% on COCO-Text, which is better than the previous state-of-the-art algorithms while taking significantly less time on average (Zhou et al. 2017).

2.2.1.2. Differentiable Binarization (DB)

Scene text detection based on segmentation-based methods is quite popular to detect scene text of arbitrary shapes. However, the post-processing of binarization is essential for segmentation-based detection, which converts probability maps into bounding boxes of text. Based on (Liao et al. 2020), a module named Differentiable Binarization (DB) is introduced which can perform binarization in a segmentation network. When a segmentation network is used in a DB module, it can set the thresholds for binarization, which helps to simplify the post-processing process and increase the efficiency of text detection. The DB module which acts as the segmentation network can adaptively establish the binarization thresholds, which improves the text detection performance. Specifically, the DB with a backbone of ResNet-18 achieves an F-score of 82.8% on the MSRA-TD500 dataset. In short, the lightweight backbone, which is the DB with real-time inference speed, the ResNet-18 method can achieve good performance on all testing datasets.

2.2.1.3. Pixel Aggregation Network (PAN)

Methods for detecting arbitrary-shaped text have been introduced, but they rarely consider the speed of the entire pipeline, which limits their deployment in the real-world environment. According to (Wang et al. 2019), an efficient and accurate arbitrary-shaped text detector, Pixel Aggregation Network (PAN) is proposed, which is equipped with a low computational-cost segmentation head and a learnable post-processing. The PAN module consists of two pipeline steps, the first which is the segmentation network's prediction of text regions, kernels, and similarity vectors. Then, using the predicted kernels, reconstruct complete text instances. A low computation-cost segmentation head that is made up of Feature Pyramid Enhancement Module (FPEM) and Feature Fusion Module (FFM) is used in PAN to remedy defects. The PAN achieves an F-score of 83.7% and 85% respectively in the CTW1500 and Total-Text datasets. In short, the PAN module accurately detects text instances of arbitrary orientation and clearly outperforms prior state-of-the-art text detectors in terms of speed and accuracy.

2.2.2. Text Recognition

Text recognition is gaining popularity due to its ability to convert text images into a string of characters or words. However, reading text in natural scenes is a challenge due to the diversity of the text shape and layout (Kulkarni and Barbadekar 2017). As a result, a few text recognition approaches are introduced to increase natural scene text recognition performance.

2.2.2.1. Pytesseract

Pytesseract is one of the python wrappers for the Tesseract OCR engine. It implements a traditional step-by-step pipeline architecture. The first image pre-processing step is with the adaptive thresholding, which results in a binary image. Then, connected component analysis is performed to extract the character outline. This method is extremely useful as it performs OCR on the image with white text on a black background [5]. The outlines are then transformed into blobs, which are then sorted into text lines. By performing character chopping and character association, the outline is converted into words. In the end, clustering and classification methods are used to perform two-pass word recognition (Zacharias et al. 2020). Based on (Gjorgjevikj and Gjoreski 2014), a comparison study was performed to identify the performance of pytesseract and Transym OCR. Pytesseract provides a better accuracy of 61% for colour images and 70% for grayscale images as compared to the Transym method. Pytesseract has a better overall performance as compared to the Transym.

2.2.2.2. Convolutional Recurrent Neural Network (CRNN)

Convolutional Recurrent Neural Network (CRNN) is the combination of the Deep Convolutional Neural Networks (DCNN) and Recurrent Neural Networks (RNN) models. CRNN possesses a distinctive advantage over the conventional neural network models. It achieves better performance on the scene text while consuming less storage size as compared to DCNN and RNN models (Yin et al. 2014). The network architecture of CRNN which consists of three components: the convolutional layers, recurrent layers, and a transcription layer. The convolutional layers which are located at the bottom of CRNN help to extract a feature sequence from the input image. A recurrent layer is built to make predictions for each frame of feature sequence output by the convolutional network. The top part of the CRNN, which is the transcription layer, is used to translate the per-frame predictions by the recurrent layer into a final label sequence (Shi et al. 2016). Performance evaluation is performed by using four commonly used benchmarks such as IIIT5k, SVT, and IC03 to achieve an accuracy of 98.7%, 96.4%, and 97.6% respectively. All the layers in CRNN have weight sharing connections, so fully connected layers are not required. Since the number of parameters used in CRNN is much lower than in other algorithms, the model is significantly smaller making it more suitable for real-world applications.

To handle text in dynamic video frames, recent literature has heavily leaned toward robust, lightweight detectors and sequence recognizers. For instance, in the ICDAR 2023 RoadText Challenge (Tom et al. 2023; Taki and Zemmouri 2023) referenced several official systems deploying DBNet-based architectures for localization and CRNN models for sequence transcription. These models have consistently proven to strike an optimal balance between processing efficiency and geometric adaptability.

2.2.2.3. RobustScanner

The attention-based encoder-decoder framework has recently achieved significant results for scene text recognition, and many variants have emerged with the improvement in recognition quality. However, it performs poorly on the random character sequence. To reduce the misrecognition, the RobustScanner module is proposed by (Yue et al. 2020) to perform text recognition by dynamically enhancing the positive clues of the decoder. Basically, the RobustScanner module consists of one position enhancement branch and one dynamic fusion module, in addition to the conventional decoder. The former is designed to improve the positional encoding capabilities of a conventional decoder by estimating glimpses with positive clues. The hybrid information of context and position are involved in the query vectors in the encoder-decoder with an attention-based framework (Yue et al. 2020). The architecture of the RobustScanner method consists of one encoder and one decoder. For the encoder, it consists of one layer of 31-layer ResNet as the backbone while for the decoder, it consists of one hybrid branch, position enhancement branch, dynamically fusing module, and prediction module. The position enhancement branch is designed to solve the alignment drift problem caused by strong contextual information. The end-to-end training of the position-aware module enforces the output feature maps correlated with positions so that the attention module can correctly output the feature glimpse at each decoding step. By comparing the RobustScanner module with the previous methods, it achieves the best performance among the IIIT5K, IC13, ICDAR 2015 (IC15), and CUTE 80 datasets. It can be observed that the RobustScanner outperforms its competitor SAR on four out of six benchmarks. It obtains an accuracy of 92.4% and outperforms SAR with only 89.6% on the challenging irregular text dataset CUTE 80. As a result, the RobustScanner can be said to be robust in both context and contextless application scenarios.

3. Methodology

In the initial stage, a custom dataset with two classes has been created which includes a total of 500 images. These are images of Malaysian road direction signs with different illumination and orientation,

obtained from the dashcam video and web resources. The next stage is to train the road direction sign detection model. The training process of the YOLOv5s with the custom dataset is run in Google Colab using Python.

The next stage is the implementation, evaluation and selection of text detection and recognition models. Three text detection models (EAST, DB, PAN) and four text recognition models (Pytesseract, CRNN, RobustScanner, ABINet) are chosen for further investigation. To select the best text detection and recognition model, 80 road sign images with different illumination and orientation are used as the input. The final chosen models will then be tested in nighttime and rainy conditions to further evaluate their performance. Analysis at this stage is performed using Visual Studio Code and Python. The final stage of the project is to implement the best model into the real-time environment. The dashcam videos are used as the input to conduct real-time analysis. Before being fed into the system, the dashcam videos will go through a pre-processing stage to improve the overall performance. Finally, the combination of road sign detection model, text detection model and text recognition model will be utilized sequentially to achieve a complete real-time text extraction system

3.1. Data Collection

Data collection plays an important role in this research because there is no available dataset for Malaysian road road sign. Therefore, a custom dataset needs to be formed to proceed with the findings. A Blueskysea B1W Mini dashcam mounted on the front windshield is used to record the video as the car travels. The video resolution of the dashcam is 1920x1080p at 30fps which is clear and sufficient for real-time application. The dataset is created by extracting the road sign and billboard images from the dashcam videos or web resources. A total of 200 dashcam videos were obtained from the dashcam. However, only 140 dashcam videos consist of road signs and billboards that can be used for image extraction. 125 road sign images and 85 billboard images with different illumination and orientation have been successfully extracted from the dashcam videos.

On the other hand, web-scraping technique is employed to obtain more road sign and billboard images for the dataset. Using this technique, Python code can be used to easily download related images from Google Chrome. As a result, a total of 160 road sign images and 130 billboard images are obtained from using web scraping. The total number of road sign and billboard images obtained from dashcam video and web resources are listed in Table 1.

Table 1 Total number of images in the custom dataset

Type of Images	Dashcam videos	Web resources
Road Sign	125	160
Billboards	85	130
Total	210	290

As a result, a custom dataset of two classes with a total of 500 images including the road sign board and billboard is created. All the bounding boxes and the class label of the road sign and billboard are obtained by using the online tool which is known as Makesense.ai. The prepared labels and the coordinates of the bounding box are exported in the form of a text file. The road sign images obtained are of different illumination and orientation.

3.2. Road sign detection model using YOLOv5s

The YOLOv5s model is chosen to perform the road sign detection as it is smaller than the other versions of YOLOv5, and able to achieve an optimal value for both accuracy and speed. The YOLOv5s model is written in Python and implemented using the Pytorch framework. Google Colab, which is an Integrated

Development Environment (IDE) that provides a Graphic Process Unit (GPU) is used to speed up the training process of YOLOv5s.

Initially, road sign detection can be considered as a binary classification task. However, since the YOLOv5s model will be implemented in the real-time environment, there will be several distractions from similar objects such as billboards that will frequently appear along the road, leading to misclassification of the road signs. To solve this problem, training the YOLOv5s model with a custom dataset of two classes which are road signs and billboard images helps to improve the overall performance of the model. Before the training process, the dataset is split into a common rate of 80-20, with 80% (400 images) for the training data and 20% (100 images) for validation data. The YOLOv5s model on Pytorch used a yaml file to access and used the image as input to the dataset. The training settings of YOLOv5s are set to 200 epochs, 640 image size, and 200 batch size.

After the training is completed, the trained weight of YOLOv5s is downloaded to perform model testing. Model testing is carried out with test images of different conditions that are not included in the training data to evaluate the performance of the trained model. When YOLOv5s is performing the detection process, a bounding box will be formed around the object with a high similarity to the road sign and billboard. As the purpose of using YOLOv5s model is to detect the road sign not the billboard, the bounding box around the billboard will not be displayed in the output. The coordinates of the bounding box around the road sign act as the boundary to crop the signboard image. Python and the OpenCV library are used to develop the code for the cropping task. The cropped road sign image will then be used as the input image to perform text detection and recognition.

3.3. Text Detection

There are three different types of text detection models which are EAST, DB and PAN. Based on Table 2, the Tesseract OCR toolbox will be used for the EAST model, the DB model is performed in the PaddleOCR toolbox and the PAN model is implemented in the MMOCR toolbox. All the models can be obtained from the Model Zoo of the respective OCR toolbox. The EAST model is 12.4MB in size while the DB and PAN models are 3.8MB and 44MB in size respectively.

Table 2 The toolbox used for each text detection model.

Toolbox	Tesseract OCR	Paddle OCR	MMOCR
Model	EAST	DB	PAN

3.4. Text Recognition

Text recognition which is also known as Optical Character Recognition (OCR) is the process of recognizing and extracting the text from the image. Four different types of text recognition models are investigated: Pytesseract, CRNN, RobustScanner and ABINet. Referring to Table 3, the Pytesseract model will be run in the Tesseract OCR toolbox while the CRNN model is run in the PaddleOCR toolbox. Besides that, the RobustScanner and ABINet models are conducted in the MMOCR toolbox. The text recognition models can be easily retrieved from the Model Zoo of each OCR tool. In addition, the Pytesseract has a model size of 14.8MB while the CRNN has a model size of 9.6MB. The model sizes of RobustScanner and ABINet models are 183MB and 141MB respectively.

In the text recognition stage, the models will only recognise text that is located within the bounding box. Following that, the recognised text will be saved in an array named listofRegText [], which will be used to display the text. One of the best text recognition models with the desired performance will be selected at the end of this section.

Table 3 The toolbox used for each text recognition model

Toolbox	Tesseract OCR	Paddle OCR	MMOCR	
Model	Pytesseract	CRNN	RobustScanner	ABINet

3.5. Performance Metric Evaluation of Model

Different performance metrics are used to evaluate different types of models. Intersection over Union (IoU) is used to evaluate object detection. Accuracy, precision and recall are used to evaluate the performance of the text detection and recognition model. F1-score, Character Error Rate (CER), and inference time are used to access the OCR performance. For real-time implementation, Frame Per Second (FPS) is used to measure performance.

3.5.1. Intersection over Union (IoU)

IoU is calculated by dividing the area of intersection of the predicted and ground truth annotations by the union of these.

$$IoU = \frac{Area\ of\ Intersection\ of\ two\ boxes}{Area\ of\ Union\ of\ two\ boxes} \quad (1)$$

IoU is obtained from the Pytorch library, and the Python programming language is used to perform this task. The performance of text detection models will be evaluated using IoU. The higher the IoU value, the better the prediction of the text detection model.

3.5.2. Accuracy, Precision and Recall

Accuracy is a metric that measures how well a model performs in all classes. It can be calculated as the ratio between the number of correct predictions and the total number of predictions. Precision is measured as the ratio of the correctly predicted positive samples to the total predicted positive samples. Precision may be viewed as a metric for how accurate or high-quality something is. Recall is the ratio of correctly predicted positive samples to the total number of positive samples. Recall is also known as sensitivity, which measures the model's ability to identify positive samples. The higher the recall score of the model, the greater the number of positive samples that are detected. Mathematically, the Accuracy, Precision and Recall can be represented with the formula below.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

where,
TP (True Positive) = Correctly predicts the positive class
TN (True Negative) = Correctly predicts the negative class.
FP (False Positive) = Incorrectly predicts the positive class.
FN (False Negative) = Incorrectly predicts the negative class

The Scikit-learn library for Python is used to calculate the accuracy, precision and recall by importing the functions of `accuracy_score()`, `precision_score()` and `recall_score()`. All the scores obtained are then converted to percentage form by multiplying by 100.

3.5.3. *F1 score*

An F1-score is a measure of a model's accuracy on a dataset. The higher the F1-score, the better the performance of the OCR model. Scikit-learn which is a Python library is used to import the function of `MultiLabelBinarizer` and `f1_score` to calculate the F1-score of the model by applying the equation below.

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

3.5.4. *Character Error Rate (CER)*

CER is a common metric of the performance of an OCR system. The output of the CER equation indicates the percentage of characters in the reference text that were incorrectly predicted in the OCR output. The greater the performance of the OCR model, the lower the CER value. Based on (Abdulkader and Casey 2009), the text recognition model with CER of 1% to 5% represents a good OCR accuracy, CER of 5% to 10% is considered as moderate OCR accuracy and CER of more than 10% indicates a poor OCR accuracy. In this work, the JiWER library is used to perform character error rate analysis. It can be represented with the equation below:

$$CER = \frac{S+D+I}{N}\ CER = \frac{S+D+I}{N} \tag{6}$$

Where, S = Number of Substitutions, D = Number of Deletions, I = Number of Insertions, and N = Number of characters in reference text (ground truth).

3.5.5. *Inference Time and Frame Per Second (FPS)*

Inference time is defined as the time taken for a deep learning algorithm to apply the trained OCR model to a new input image. Python timeit() is used to measure the inference time taken by the given code snippet. The complexity of the network and the number of layers can influence the inference time. In general, the inference time increases with the network size and complexity. The shorter the inference time, the more efficient the OCR model is.

A Frame per second (FPS) is used to measure the display performance of video captures. FPS can be described as the number of images that are consecutively displayed in a second. Therefore, the higher the FPS, the smoother the video motion appears. In this research, FPS is used to evaluate the video quality of the real-time text extraction system.

3.6. *Realtime Implementation*

Fig. 1 outlines the system's real-time implementation. To meet strict real-time requirements, ten input dashcam videos undergo a critical preprocessing stage that accelerates inference. This step optimizes processing speed by downscaling the video resolution and cropping out the bottom portion of the frame.

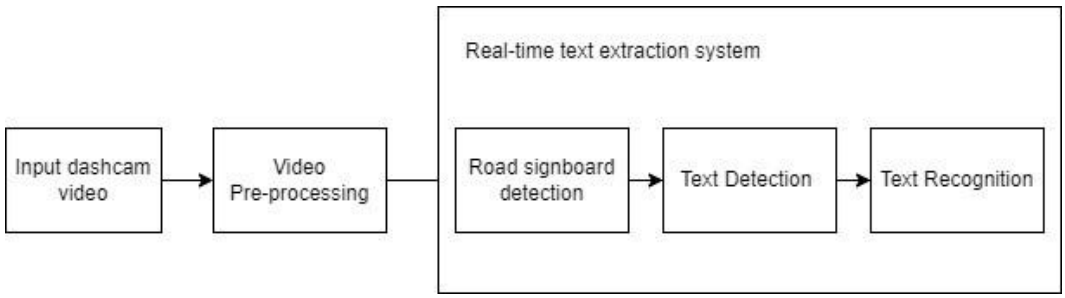


Figure 1. Flowchart of real-time implementation process

With the 1920x1080p resolution of the dashcam video, it is relatively large for the system to process. Thus, the video is downscaled with a scaling factor of 0.5. The before and after downscaling of the dashcam video is shown in Fig. 2. This can be done by using the `cv2.resize()` function of the OpenCV library in Python.

Figure 2. (a) Frame before downscaling (b) Frame after downscaling

Since the road sign will only appear in the upper part of the dashcam video, cropping the one-third portion from the bottom part of the video will enhance the overall performance of the system. This can be done easily by using OpenCV library. Fig. 3 illustrates the before and after cropping process of the road sign. By applying both video pre-processing methods into the system, the resolution and size of the video are reduced, resulting in a reduction of processing time.

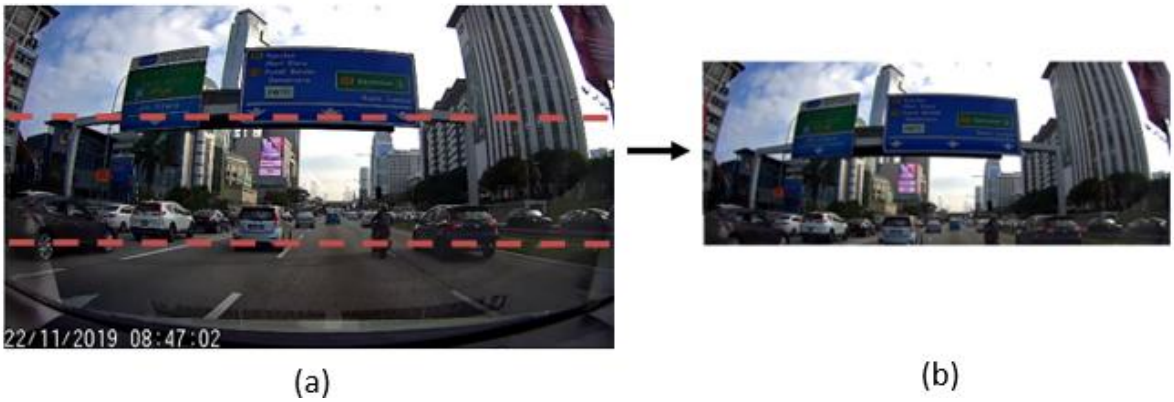


Figure 3. (a) Frame before cropping (b) Frame after cropping.

The dashcam videos at 35 frames per second are fed into the system and begin processing each frame of the video. The dashcam videos are recorded at a typical driving speed, as according to traffic during daytime and night-time. The dashcam videos undergo the video pre-processing process which includes the downscaling and cropping the bottom part of the video. After that, the road sign detection is performed by using the pre-processed video. However, the road sign detection is intended to assist in not only detecting, but also cropping the area of the road sign to speed up the text extraction process. The road sign detection model will filter out the detected road sign image with the width lower than a threshold value. The width of the road sign is calculated by finding the difference between the x-coordinate of two points while the threshold value is set to 10% of the frame's resolution. This is a crucial step to filter out the weak predictions that will affect the performance of the text recognition model. If the width of the road sign fulfils the requirement, the model will crop the area of the road sign to perform text detection. Bounding boxes will be formed around the text of the road sign and will proceed to the text recognition stage. The detected text obtained from each frame of the video will be saved and displayed as the output. In short, if the system is capable of processing text extraction right after each frame of the video, this clearly demonstrates a real-time text extraction system. The whole process of the real-time implementation will be performed on a i7-1154G7 with NVIDIA GeForce MX350 GPU laptop using Python.

4. Result and Discussion

In this section, the results of training and validation of YOLOv5s will be discussed. A total of 80 road sign images are used to validate the performance of text detection and recognition models. The text detection model that obtains the highest IoU and fastest inference time will be chosen as the best model. Moreover, the model that achieves the desired performance in different conditions will be chosen as the best text recognition model. The selected model will then be used to perform in the worst-case condition to further investigate its performance and efficiency. Several performance metrics will be used to evaluate the performance of the models. Lastly, the selected models will then be implemented in the real-time environment to perform analysis.

4.1. Road sign Detection

The YOLOv5s model is trained by using a custom dataset with two classes: road sign and billboard. The default training settings of YOLOv5 model are used to observe the performance of the baseline model and spot the area of improvement if needed. The default training settings are set to 100 epochs, 416 of image size and 32 of batch size. The performances of the trained model are evaluated using box loss, objectness loss, classification loss, precision, recall, and mAP (mean Average Precision) when IOU is at 0.5 and 0.95.

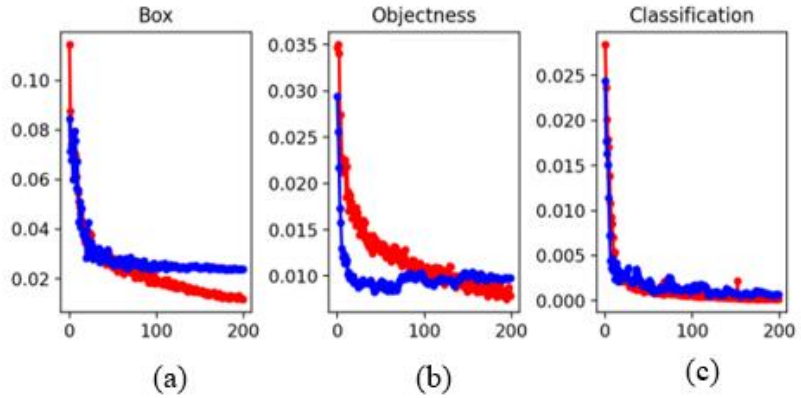


Figure 4. Graph of (a) box loss, (b) objectness loss, (c) classification loss of YOLOv5s training and validation process

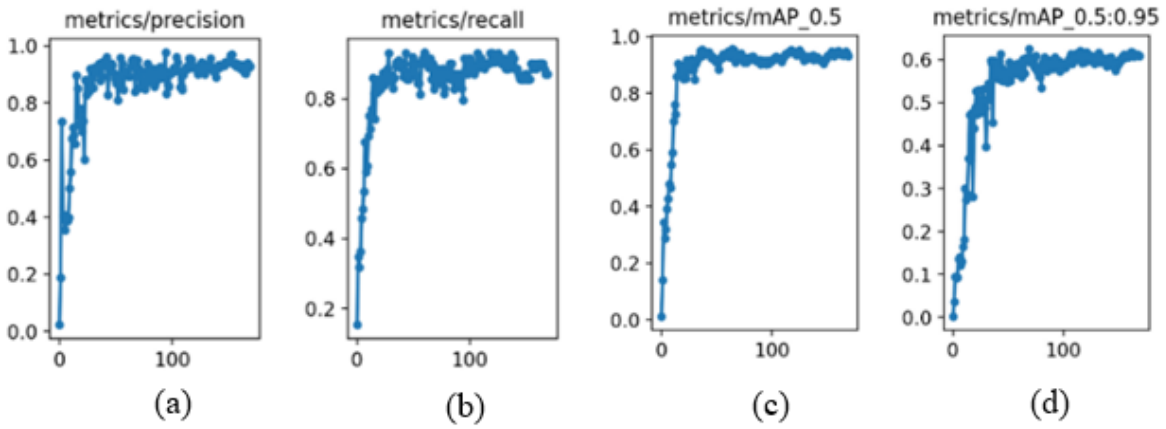


Figure 5. Graph of (a) Precision, (b) Recall, (c) mAP@0.5 and (d) mAP@0.95 for YOLOv5s training and validation progresses

In Fig. 4, the red line in the plot represents the training loss while the blue line represents the validation loss of the YOLOv5s model. The plot validation loss is visibly reduced to a point of stability, with just a little gap between it and the training loss. As a result, a good fit model was defined by the plot of training loss and validation loss. Moreover, the trained YOLOv5s model has a validation precision score of 97%, a recall score of 96%, and mAP scores of 0.937 for IOU at 0.5 and 0.625 for IOU at 0.95 as shown in Fig. 5. With a precision score of 97% and a recall score of 96%, this proves that the YOLOv5 by itself achieves the desired result without any optimization method integrated into it.

Next, 100 road sign and billboard images are used for the validation of each class. It can be observed from the confusion matrix in Fig. 6 that the model is able to detect almost all the road sign correctly with a true positive value of 0.98. This clearly demonstrates the high efficiency of the model in detecting road sign in different orientations and illumination.

To further verify the performance of the trained YOLOv5s model, eight test images that are non-duplicated with the training images are used to perform model testing. The model testing is conducted by setting the confidence threshold to 0.5. Fig. 7 show the results of test images for road sign detection and billboard detection during daytime and night-time.

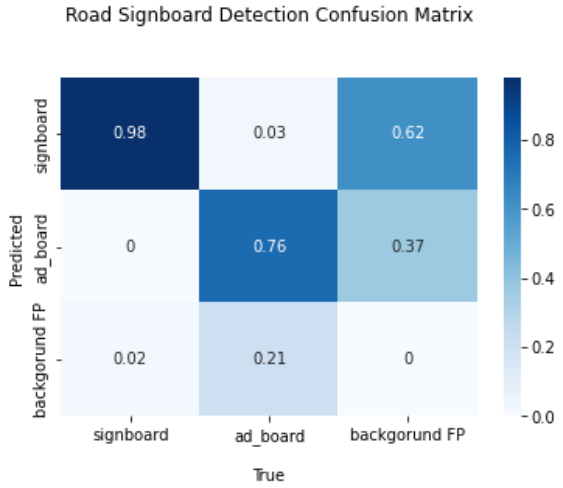


Figure 6. Confusion matrix on road sign detection using YOLOv5s.

Based on the results in Fig. 7, the YOLOv5s model can detect all the road signs and billboards accurately for both day and night-time conditions. The trained YOLOv5s model with two different classes which are road sign and billboard clearly improved the accuracy of the system and eliminated the misclassification problem. This demonstrates the robustness and effectiveness of the trained YOLOv5s model. The YOLOv5s model is lightweight, it has great potential to detect road signs in real-time environments.

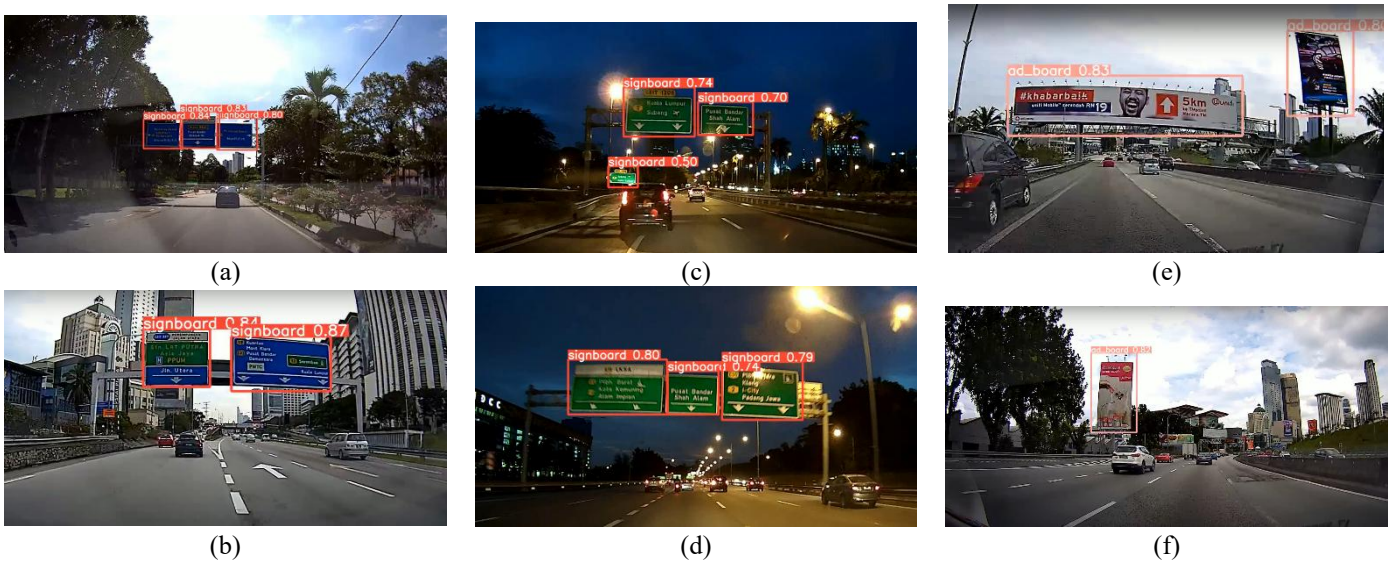


Figure 7. Results of YOLOv5s model for signboard detection during daytime (a, b), during nighttime (c, d), and billboard detection during daytime (e, f)

4.2. Text Detection

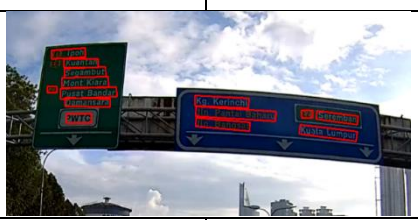
A total of 80 road sign images with different orientations and illumination are used to validate the performance of three different text detection models which are EAST, DB and PAN. The bounding box will form if the text is being detected. To visualize the performance of each algorithm, three types of input

images of different orientation and illumination are sampled: Image Type 1, the road sign image with approximately horizontal text, Image Type 2 the road sign image has an arbitrary-oriented text and Image Type 3 is an image of a darker road sign with approximately horizontal text.

Referring to Table 4, for input Image Type 1, the PAN model underperformed because some of the texts were not detected when compared to the other models. From the results, the DB model obtained the highest IoU of 0.8330 and the fastest inference time of 0.157s as compared to the EAST and PAN models. For input Image Type 2, the DB model still outperforms the EAST and PAN models with an inference time of 0.8065 and an inference time of 0.161s. For input Image Type 3, the road sign image has an approximate horizontal text that is captured in the daytime, but the signboard appears to be darker as the back of the signboard is exposed to the sun. However, the DB model is still able to perform well in this condition by obtaining an IoU of 0.7845 and an inference time of 0.156s. In these examples the DB model outperforms the other models in all three scenarios.

Fig.8 illustrates the comparison bar chart of the text detection models. Average IoU and inference time are used as the metrics to analyze the performance of the text detection models. As mentioned previously, the value of IoU which is greater than 0.8 indicates the good efficiency of the model.

Table 4 Results of different text detection models (EAST, DB and PAN) for Input Image Type 1, 2 and 3

	EAST		**DB**		**PAN**	
Image Type 1						
	IOU: 0.8064	Inference: 0.325	IOU: 0.8330	Inference: 0.157	IOU: 0.7155	Inference: 0.216
Image Type 2						
	IOU: 0.7934	Inference: 0.426	IOU: 0.8065	Inference: 0.161	IOU: 0.7198	Inference: 0.233
Image Type 3						
	IOU: 0.8011	Inference: 0.424	IOU: 0.7845	Inference: 0.156	IOU: 0.7397	Inference: 0.174

Based on the results obtained, the EAST model has an average inference time of 0.4217s which is double the average inference time of the DB model which is 0.1938s. Since the model size of the DB model is only 3.8MB, it will further accelerate the process of text detection. In addition, the average IoU of the EAST and PAN models, which are 0.7103 and 0.7652 respectively, does not fulfil the requirement of 0.8. Therefore, the EAST and PAN models only achieve moderate efficiency as compared to the DB model. The average IoU of the DB model remained the highest, which is 0.8238 and with the lowest inference time of 0.1938s. In a nutshell, the DB model achieves the desired performance with the highest IoU and the fastest inference time based on the 80 road sign images. Therefore, the DB model was chosen as the best text detection model, which will be combined with the text recognition models in the next stage.

4.3. Text Recognition

The chosen DB model is combined with 4 different text recognition models to further perform the text recognition. The input images are the same as in the text detection stage. The text recognition models will recognise the text within the bounding box of the road sign and display the text as the output. A sample output of text recognition using all four models onto the Image Type 2 is shown in Table 5. For each model, the numbers indicate the confidence level of the text recognition result.

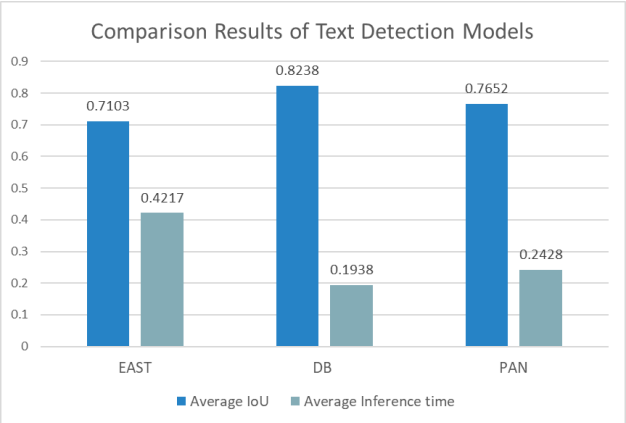


Figure 1. Comparison bar chart of text detection models with 80 input images

Table 5 Text Recognition Result

Text Detection Model (DB)	Text Recognition Model			
	Pytesseract	**CRNN**	**RobustScanner**	**ABINet**
	1: padang. jawa 0.887 2: i-city 0.813 3: klang 0.871 4: exitison) 0.850 5: bandar. diraja 0.894 6: kualan. lumpur 0.874 7: subang 0.851	1: XIT1306 0.980 2: Padang Jawa 0.956 3: 1-City 0.947 4: Klang 0.992 5: Bandar DiRaja 0.976 6: Subang 0.998 7: Kuala Lumpur 0.960	1: Kuala 0.881 2: lumpur 0.877 3: Subang 0.862 4: Bandar 0.888 5: diRaja 0.911 6: KLANG 0.882 7: City 0.866 8: Padang 0.845 9: Jawa 0.861 10: EXIT 0.832 11: 1306 0.785	1: kuala 0.881 2: lumpur 0.877 3: subang 0.862 4: bandar 0.888 5: diraja 0.911 6: klang 0.882 7: city 0.866 8: padang 0.845 9: jjawa 0.861 10: rexit 0.832 11: tod 0.785

Table 6 Result of the text detection models with 80 input images.

Parameter	Model			
	Pytesseract	**CRNN**	**RobustScanner**	**ABINet**
Average CER (%)	12.34	**5.85**	9.01	11.89
Average accuracy (%)	78.97	**97.84**	88.36	82.55
Average precision (%)	76.08	**93.78**	90.38	81.78
Average recall (%)	78.68	**97.68**	91.54	82.78
Average F1-score (%)	77.36	**96.24**	92.07	80.43
Average inference time (s)	0.562	**0.324**	0.658	0.581

Table 6 gives the result of the text recognition models using 80 input images. Based on the results obtained, the average accuracy, precision, recall and F1-score of CRNN model achieved the highest among the other models. To obtain good OCR accuracy, the CER value of the model must be around 1-5%. Based on this requirement, only CRNN meets the criterion, with an average CER value of 5.85%.

The CER of the text recognition model plays an important role in selecting a model for the text extraction system. This is because CER indicates the percentage of characters in the text that are incorrectly predicted in the output. The lower the CER value, the better the performance of the model. The CRNN model also has the fastest inference time of 0.324s as compared to Pytesseract, RobustScanner and ABINet. As the model size of CRNN is small with only 9.6MB, it is suitable for real-time application and able to run at a real-time speed. Since the goal of the research is to implement the text extraction into the real-time environment, the chosen model needs to achieve the optimal value of CER, accuracy, and inference time. Thus, the combination of DB and CRNN are selected as the best text detection and recognition models.

4.3.1. Performance Evaluation in Worst-Case Conditions

In this section, DB and CRNN models are used as the text detection and recognition models to perform in the worst-case conditions to further investigate their performance and efficiency. They are tested in two worst-case scenarios which are night-time and rainy-day conditions. Fig 9 shows the 2 input images used for nighttime text recognition analysis and Table 7 gives the respective result.

Image	Text detection	Text recognition
1		1: GXIVATNOO 0.629 2: Kuala Lumpur 0.910 3: Subang 0.983 4: Pusat Dandar 0.930 5: Shah Alam 0.957

2		1: EU-LKSA 0.765 2: 130 0.600 3: Plbh Utars 0.737 4: PIb arat 0.718 5: Klang 0.963 6: Keta Kemuning 0.824 7: 2 0.964 8: Pusat Dandar 0.963 9: i-City 0.925 10: Alam Impian 0.902 11: Shah 0.988 12: Alam 0.990 13: Padang Jaws 0.927

Figure 2. Two nighttime images tested and the confidence level of the text recognition result.

It is a challenging task to perform text detection and recognition at night. This is because the low light exposure at night will produce a lot of noise on the image and results in a poor image quality. As we mentioned earlier, the model must achieve a CER around 1-5% to achieve good OCR accuracy. Based on the results obtained, both images have a high CER which is 12.5% and 26.31% respectively, indicating a poor quality of OCR.

Table 7 Performance metrics in night-time condition

Parameter	Night Image 1	Night Image 2
CER (%)	12.50	26.31
Accuracy (%)	72.00	73.69
Precision (%)	69.35	71.42
Recall (%)	71.14	66.47
F1-score (%)	70.23	68.86
Inference time (s)	0.265	0.327

In addition, the accuracy, precision, recall and F1-score of DB and CRNN models at night-time are reduced approximately 20% as compared to the overall performance of DB and CRNN models. However, the inference time obtained from the model based on both night-time images are still considered fast, which is 0.265s and 0.327s respectively. Overall, the performance of DB and CRNN models during the night-time condition is not that good. Therefore, improvements are required to improve the OCR accuracy as well as the performance of DB and CRNN models.

Fig. 10 shows the selected rainy-day images used and Table 8 gives the results obtained when these two rainy-day images are used as input. On a rainy day, the signboard image captured by the dashcam will be blurred. This is because the rain will scatter the lights and distort the visual scene.

Table 8 Performance Matric in Rainy Day Conditions

Parameter	Rainy Image 1	Rainy Image 2
CER (%)	18.18	11.00
Accuracy (%)	75.0	79.33
Precision (%)	72.52	76.29
Recall (%)	74.05	75.37
F1-score (%)	73.28	75.83
Inference time (s)	0.204	0.184

Based on the results obtained, both the road sign images obtained a high CER with 18.18% and 11% respectively, which indicates poor OCR accuracy. Although both road sign images obtained a fast inference time, the accuracy, precision, recall, and F1-score are approximately 70% which indicates an

unsatisfactory performance. In short, it can be concluded that DB and CRNN models have poor OCR accuracy and weak performance during rainy days.

4.4. Real time Implementation

Based on the results obtained from the previous sections, the proposed models used for each stage of real-time text extraction system are shown in Table 9.

Table 9 Model used in each stage of real-time text extraction system.

Stage	Model
Road sign Detection	YOLOv5s
Text Detection	DB
Text Recognition	CRNN

In a real-time system, the task must be completed within a specified time frame. A large model size takes a longer time to process, which influences the model processing time. Therefore, the model size of proposed models which is 27 MB is small as compared to the other models, making it easier to perform real-time detection and recognition. However, for our research, the real-time text extraction system will only be implemented in daytime conditions, as some improvement needs to be carried out in night-time and rainy-day conditions. The entire real-time implementation is carried out using a laptop equipped with an NVIDIA GeForce MX350 GPU. A total of ten dashcam videos are fed into the real-time text extraction system to analyze the results. The text from each frame of the video will be extracted by the system and displayed at the bottom of the dashcam video.

Image	Text detection	Text recognition
1		1: PERSIARAN KAYANGAN 0.966 2: Parstaran 0.916 3: TengluAmpuas 0.848 4: Seksyen 15-36 0.881 5: Puchong 0.995 6: K tompur 0.823 7: Kiang 0.818

2		1: LEBUHRAYA PLUS 0.971 2: Shah Alam 0.940 3: Klang 0.939 4: E1 Ipoh 0.821 5: KLIA 0.966 6: E2Johor Bahru 0.938

Figure 10. The two rainy images tested and the confidence level of the text recognition result.

One out of 10 dashcam videos are used to further explain and help visualize the real-time implementation. In Fig. 11, several frames are extracted from the dashcam video, and the detected text of each frame is shown at the bottom of the video with a black background. The ground truth text of each signboard is used to compare with the detected text. The detected text with the lowest CER will be the best result obtained from the system.

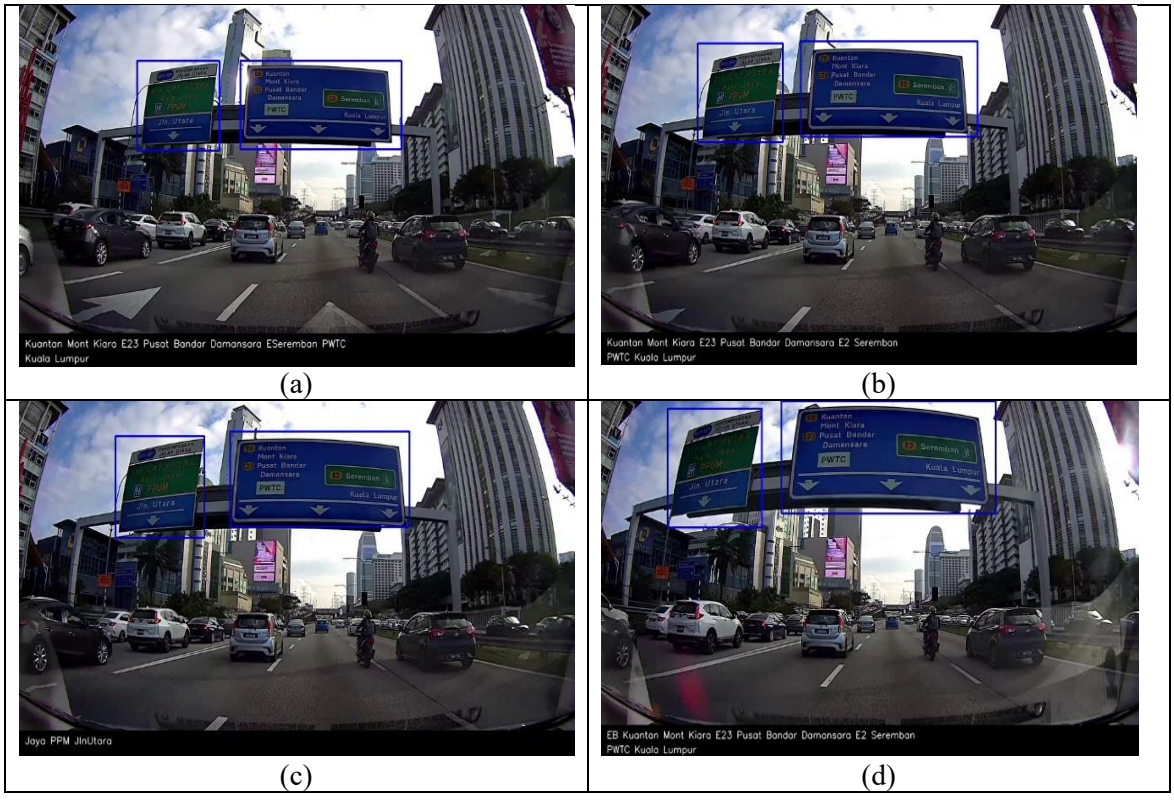


Figure 11. Several consecutive frames from (a) to (d) extracted from the dashboard camera video.

Fig. 11 demonstrates that as the vehicle approaches the road signs (from left to right), the target size increases. This enhanced proximity improves character clarity, yielding progressively more accurate extraction results as shown below the frames.

As the video plays, several frames that contain the same road signboard will be captured and processed. In each frame the CER of the detected text will be calculated. The lowest CER of the detected text will be given as the outcome of this experiment as shown in Table 10. The lowest CER of 7.69% and 12.07% is obtained for signboard 1 (left) and signboard 2 (right) respectively is obtained from the dashcam video in Fig. 11. This particular result is achieved with an FPS of 6.28.

By using the ten dashcam videos as the reference, the text extraction system completed the whole process of road signboard detection, text detection and text recognition at a speed of close to 6.5 FPS on a laptop which is equipped with NVIDIA GeForce MX350 GPU.

Table 10 Results of realtime text recognition using recorded dashcam video

Dashcam Video Realtime Implementation		
Parameter	**Signboard 1 (Left)**	**Signboard 2 (Right)**
Ground Truth Text	Kuantan, Mont Kiara, Pusat Bandar, Damansara, PWTC, E8, E23, E2, Seremban, Kuala Lumpur	Persimpangan, Jalan Utara, EXIT 227, Stn. LRT PUTRA, Asia Jaya, PPUM, Jln. Utara
Detected Text	Kuantan, Mont Kiara, Pusat Bandar, Damansara, PWTC, E23, E2, Seremban, Kuala Lumpur	Persimpangan, Jalan Utara, EXT27, LRT PUTRA, Asia Jaya, PPUM, Jln Utara
CER (%)	7.69	12.07
Average FPS	6.28	

5. Conclusion and Future Works

This project aims to develop a real-time text extraction system that can extract the text from Malaysian road direction sign using dashcam. A custom dataset is created which includes the images of the road sign with various illumination and orientation. The real-time text extraction system can be divided into three main stages which are the road sign detection, text detection and text recognition. For road sign detection, the YOLOv5s model is trained with the custom dataset and managed to achieve a precision score of 97%. For text detection stage, the DB model is selected as the best model with the highest IoU of 0.8238 and the fastest inference time of 0.1938s. For the text recognition stage, the CRNN model is selected with the lowest CER of 5.85%, an accuracy of 97.84% and the shortest inference time of 0.324s. The goal of this project is to implement the text extraction system into the real-time environment, so the model selected needs to be lightweight enough to process the text. The DB and CRNN models can achieve optimal performance in terms of accuracy and speed.

The proposed models YOLOv5s, DB and CRNN are combined to run sequentially in the real-time system. However, the real-time text extraction system is applicable only for daytime conditions, as further optimization of proposed models is needed for night-time and rainy conditions. Dashcam videos are used as the input to perform real-time system analysis. The proposed models completed each stage of the real-time text extraction system with a speed of 6.5 FPS, with an accuracy of 97%, on a laptop that is equipped with an NVIDIA GeForce MX350 GPU. Compared to similar work by (Shao et al. 2018), their real-time traffic sign (symbols) detection and recognition system achieved an FPS of 6.3 and accuracy of 91%. This clearly shows that our system outperforms the existing system with an extra feature of road direction

sign detection and extraction. In short, the proposed model is feasible to perform text extraction in a real-time environment by processing the text extraction right after each frame of the video.

Several modifications can be pursued to enhance the system's real-world dep8loyability. Training the models on larger, more diverse datasets will improve generalizability and classification accuracy, while structural optimizations are needed to handle adverse environments like nighttime and rainy conditions. Additionally, introducing post-processing spelling and linguistic correction layers can further reduce the system's Character Error Rate (CER). Upgrading the front-end detector to newer architectures, such as YOLOv8 (Wang et al. 2025), will specifically improve the detection of small, distant, or degraded traffic signs. Ultimately, transitioning from Connectionist Temporal Classification (CTC) models like CRNN to modern transformer-based recognition paradigms, as reported in (Liu et al. 2025), constitutes the primary future direction of this work.

Declaration

Data Availability

Data sets generated during the current study are not available to the public as it is ongoing student research. However, the database can be easily generated, and all models used are openly accessible to the public.

Fundings

The authors acknowledge the financial support provided by the Universiti Malaya Living Lab Towards Just Net Zero under Grant No. LL2024JNZ002.

Conflicts of Interests/Competing Interests

The authors declare that they have no conflict of interest.

References

Abdulkader, A., and M. R. Casey. 2009. "Low Cost Correction of OCR Errors Using Learning in a Multi-Engine Environment." In 2009 10th International Conference on Document Analysis and Recognition.

Agrawal, Anand. 2017. "Sign Board Detection and Information Extraction for MAVI (Mobility Assistant for Visually Impaired) Project." Thesis, Indian Institute of Technology Delhi. http://www.cse.iitd.ac.in/mavi/docs/Thesis_Anand.pdf.

Chincholkar, Y., and A. Kumar. 2019. "Traffic Sign Board Detection and Recognition for Autonomous Vehicles and Driver Assistance Systems." ICTACT Journal on Image and Video Processing 9: 1954–1959. https://doi.org/10.21917/ijivp.2019.0277.

Geethapriya, S., N. Duraimurugan, and S. Chokkalingam. 2019. "Real-Time Object Detection with Yolo." International Journal of Engineering and Advanced Technology (IJEAT) 8 (3S).

Gjorgjevikj, D., and H. Gjoreski. 2014. "Optical Character Recognition Applied on Receipts Printed in Macedonian Language." https://doi.org/10.13140/2.1.1632.4489.

Iqbal, A. 2020. "Obstacle Detection and Track Detection in Autonomous Cars." IntechOpen. https://doi.org/10.5772/intechopen.89917.

Khalid, S., J. H. Shah, M. Sharif, F. Dahan, R. Saleem, and A. Masood. 2024. "A Robust Intelligent System for Text-Based Traffic Signs Detection and Recognition in Challenging Weather Conditions." IEEE Access 12: 78261–78274. https://doi.org/10.1109/ACCESS.2024.340104.

Kulkarni, C. R., and A. B. Barbadekar. 2017. "Text Detection and Recognition: A Review." International Research Journal of Engineering and Technology (IRJET) 4.

Liao, M., Z. Wan, C. Yao, K. Chen, and X. Bai. 2020. "Real-Time Scene Text Detection with Differentiable Binarization." In Proceedings of the AAAI Conference on Artificial Intelligence.

Liu, Z., R. Song, K. Li, and Y. Li. 2025. "From Detection to Understanding: A Systematic Survey of Deep Learning for Scene Text Processing." Applied Sciences 15 (17): 9247. https://doi.org/10.3390/app15179247.

Lu, M., Y. Mou, C.-L. Chen, and Q. Tang. 2021. "An Efficient Text Detection Model for Street Signs." Applied Sciences 11 (13). https://doi.org/10.3390/app11135962.

Nepal, U., and H. Eslamiat. 2022. "Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs." Sensors 22 (2): 464.

Patel, C., A. Patel, and D. Patel. 2012. "Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study." International Journal of Computer Applications 55: 50–56. https://doi.org/10.5120/8794-2784.

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. "You Only Look Once: Unified, Real-Time Object Detection." In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 27–30 June 2016.

Rolison, Jonathan J., Shai Regev, Salissou Moutari, and Aidan Feeney. 2018. "What Are the Factors That Contribute to Road Accidents? An Assessment of Law Enforcement Views, Ordinary Drivers' Opinions, and Road Accident Records." Accident Analysis & Prevention 115: 11–24. https://doi.org/10.1016/j.aap.2018.02.025.

Sabu, A. M., and A. S. Das. 2018. "A Survey on Various Optical Character Recognition Techniques." In 2018 Conference on Emerging Devices and Smart Systems (ICEDSS), 2–3 March 2018.

Shao, F., X. Wang, F. Meng, T. Rui, D. Wang, and J. Tang. 2018. "Real-Time Traffic Sign Detection and Recognition Method Based on Simplified Gabor Wavelets and CNNs." Sensors 18 (10). https://doi.org/10.3390/s18103192.

Shi, B., X. Bai, and C. Yao. 2016. "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition." IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (11): 2298–2304.

Swetha, S., and P. Sivakumar. 2021. "SSLA Based Traffic Sign and Lane Detection for Autonomous Cars." In 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 25–27 March 2021.

Taki, Youssef, and Elmoukhtar Zemmouri. 2023. "Scene Text Recognition for Text-Based Traffic Signs." In Advances in Intelligent Traffic and Transportation Systems: Proceedings of the 6th International Conference on Intelligent Traffic and Transportation (ICITT), 25–27 September 2022.

Tom, G., M. Mathew, S. Garcia-Bordils, D. Karatzas, and C. Jawahar. 2023. "ICDAR 2023 Competition on RoadText Video Text Detection, Tracking and Recognition." In Document Analysis and Recognition - ICDAR 2023, edited by G. A. Fink, R. Jain, K. Kise, and R. Zanibbi, 14188: 529–544. Springer, Cham. https://doi.org/10.1007/978-3-031-41679-8_35.

Wang, G., P. Jin, Z. Qi, et al. 2025. "Traffic Sign Detection Method Based on Improved YOLOv8." Scientific Reports 15: 19385. https://doi.org/10.1038/s41598-025-03792-0.

Wang, H., C. Pan, X. Guo, C. Ji, and K. Deng. 2021. "From Object Detection to Text Detection and Recognition: A Brief Evolution History of Optical Character Recognition." Wiley Interdisciplinary Reviews: Computational Statistics 13. https://doi.org/10.1002/wics.1547.

Wang, W., E. Xie, X. Song, Y. Zang, W. Wang, T. Lu, G. Yu, and C. Shen. 2019. "Efficient and Accurate Arbitrary-Shaped Text Detection with Pixel Aggregation Network." In Proceedings of the IEEE/CVF International Conference on Computer Vision.

Wang, Z., Y. Wu, L. Yang, A. Thirunavukarasu, C. Evison, and Y. Zhao. 2021. "Fast Personal Protective Equipment Detection for Real Construction Sites Using Deep Learning Approaches." Sensors 21 (10). https://doi.org/10.3390/s21103478.

Yin, X., X. Yin, K. Huang, and H. Hao. 2014. "Robust Text Detection in Natural Scene Images." IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (5): 970–983. https://doi.org/10.1109/TPAMI.2013.182.

Yue, X., Z. Kuang, C. Lin, H. Sun, and W. Zhang. 2020. "Robustscanner: Dynamically Enhancing Positional Clues for Robust Text Recognition." In European Conference on Computer Vision.

Zacharias, E., M. Teuchler, and B. Bernier. 2020. "Image Processing Based Scene-Text Detection and Recognition with Tesseract." arXiv, arXiv:2004.08079.

Zhou, X., C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. 2017. "East: An Efficient and Accurate Scene Text Detector." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.