VEHICLE LICENSE PLATE CHARACTER RECOGNITION

Perivitta Rajendran, Lou Jia Yu, Yee Wen San, Dharisini Kanesamoorthy

Students of Multimedia University Cyberjaya in Faculty of Computing and Informatics

ABSTRACT

Transportation plays a big role in human life no matter transporting goods or humans. Many countries had set up some traffic cameras on major roads such as highways to observe vehicular traffic for safety purposes. Therefore, it is essential to identify the vehicle license plate effectively. In this paper, a comparison of three machine learning techniques is proposed to evaluate the accuracy of recognizing the vehicle license plate character after performing the license plate detection and character segmentation.

Index Terms— License plate recognition, character segmentation, Optical Character Recognition (OCR), Random Forest Classification, Artificial Neural Network (ANN).

1. INTRODUCTION

There are always vehicle-related crimes and traffic jams occurring all over the world. License plate detection or Automatic Number Plate Recognition (ANPR) or some would even call it License plate Recognition (LPR) is something that is constantly seeking for monitoring and easing investigation that is related to vehicles. In addition to that, LPR also helps to provide better security, ticketless parking, and even plunge into the research of smart cities that use LPR to understand traffic. LPR is defined as the capability of capturing an image or a digital video of a license plate and transforming it into optical data that allows us to gather digital information from it.

Through the detection of license plates were able to derive and plate different types of information such as the type of vehicle, where it originated from, the driver's information, and finally, the records of complaints and warnings received by the vehicle. There are many attempts on creating the optimal LPR but there are different types of license plates being created all over the world, with different designs and state codes.

This project will be looking into the optimal method for license plate character recognition by implementing different algorithms and fine-tuning them to produce the best accuracy. This project will be using two datasets one that is sourced from Kaggle which has about 433 cars along with their license plate and the file size is about 212.88MB. The other set of data was collected from the members capturing car images in Malaysia. It contains 120 images of cars with Malaysian license plates.

2. RELATED WORK

A review has been done to study the related work on license plate character recognition which has been performed by several researchers in the past few years.

M. M. AlyanNezhad et al, 2017 [1] has proposed a fusion of Gaussian Filtering and Bayesian Network on detecting the license plate in complex scenes. Conversion of the original image to a gray-scale image is crucial before stepping to the next step. A histogram equalization method is performed to enhance the images for those that are in low contrast and low brightness. Because the density of the vertical edge is high in the license plate region, Sobel Operator is used to calculate the edge and then apply 2D Gaussian Filter to the vertical edge image. The images are then converted to binary form by performing the Open and Close Morphological Operators, and lastly applying a matched filter before implementing the Bayesian Network.

A. Menon et al, 2018 [2] have presented an Automatic Number Plate Recognition (ALPR) system to detect and recognize multiple license plates in still images. The machine learning techniques that are involved in this system are Support Vector Machine (SVM) and Artificial Neural Network (ANN). Firstly, after the images were converted into grayscale, a Sobel filter is used to detect edges and then threshold the images using OTSU's method. Closing Morphological Operation and Flood Fill algorithms are implemented to separate the interest regions from their background. The authors next utilized the SVM technique to predict if the detected regions consist of a license plate, if it consists of a license plate, the ANN technique is applied to predict the segmented character.

According to Koval et al, 2003 [3], the authors introduce an approach for Number Plate Recognition (NPR) using morphological operation and Sobel edge detection method. Dilation and erosion morphology was applied to recognize the area of the license plate and Sobel edge operations are made to compute the threshold. Every character and number presented in the license plate was segmented by using the bounding box method. After segmentation, the authors employed a template matching technique to recognize characters and numbers on the license plate. The matching procedure is done on a pixel-by-pixel premise.

P. Faizal et al, 2018 [4] had proposed using image fusion, neural networks, and threshold techniques for license plate recognition to identify vehicles. To identify the license plate of a vehicle, the author used an iterative thresholding operation. The rectangle region of the license plate is being executed and inverted. Besides, they also applied a median filter to the inverted license plate image to eliminate noise. In order to obtain single characters from the license plate, the filtered image undergoes a segmentation process whereby each character region was extracted based on the maximum and minimum height and width parameters. A feed-forward neural network was used for the license plate recognition system to predict each segmented character from the license plate.

According to R. V. Mario et al, 2019 [5], they have developed a system that recognizes characters from images using Computer Vision techniques. This system was implemented in many fields like parking lots, private and public entrances, toll gates, theft control, and checking the authentication of a vehicle. The procedure followed in this paper is, first capturing images from the camera then loading that into the system, preprocessing is done using the OpenCV library. It involves the change of color, change of background illumination, enhancing contrast, saturation, removing background noise, application of contours. Image thresholding was used for segmentation and the number plate localization to find the region of interest. Then, the implementation of Attention OCR, a deep learning model is used to recognize the characters from an image. Overall, OCR tends to perform well in identifying street signs and in Natural Language Processing which identifies the characters according to the sentence formation technique.

In this research Akhtar et al, 2020 [6], they have proposed an embedded system that recognizes the number plate of the vehicle. It was efficient in detecting the number plate in noisy as well as in low illumination and also within the required time frame. This paper proposed a number plate recognition method by processing the vehicle's rear or front image. After an image is captured, the processing is divided into four steps which are preprocessing, number plate localization, character segmentation, and character recognition. Preprocessing enhances the image for further processing, number plate localization extracts the number plate region from the image, character segmentation separates the individual characters from the extracted number plate, and character recognition identifies the optical characters by using a random forest classification algorithm. Random forest works best among several classification methods because it is based on an ensemble of classifiers, precisely decision trees.

Abdullah et al, 2009 [7] made an objective to create an enhanced procedure of object detection and enhance feature extraction. They have expanded a procedure based on Cluster Run Length Smoothing Algorithm (CRLSA), an improved Geometrical Feature Topological Feature Analysis(eGFTA), and finally Support Vector Machine (SVM). The performance of this prototype was then compared to a Multi-layer Perceptron that was trained along with backpropagation (MLP-BP). Before detection, segmentation, and classification a homomorphic filter is applied to enhance images that are captured and then thresholded and labeled to perform segmentation. The accuracy of the methods was plotted and compared with MLP-BP.

Shi X et al, 2005 [8] have suggested an LPR method that extracts color and shape information from the vehicle license plate, and later the ROI is thresholded and segmented vertically and horizontally. The RGB model was utilized to perform the classification of pixels into 13 color categories. A bimodal histogram was used to perform segmentation. Based on a template, a minimum Euclidean distance is computed and further confusing characters such as ‘8’ and ‘B’ were processed further. Finally, the validity is checked by machine and manual.

3. Approach

This section explains the methodology of the project.

3.1 Image Preprocessing

As mentioned in the introduction this project will be using two datasets, one sourced from Kaggle and another data was collected by the members.

Interference in the original input images like noise, distortion, size, and quality of the image will affect the recognition performances. To minimize these interference factors, preprocessing is required on the input image so that the quality of the image will be improved and save up computational time.

Figure 3.1 shows the original car image. First, color images to the grayscale conversion will be performed. It is a major step to speed up and ease the following imaging processes. After that, bilateral filtering will be applied to the gray image to remove the edge of noise surrounding the license plate, while preserving the edges of characters in the image sharp. Figure 3.2 displays the sample gray image after the bilateral filter was applied. This step can also ensure the algorithm avoids unrelated background details.

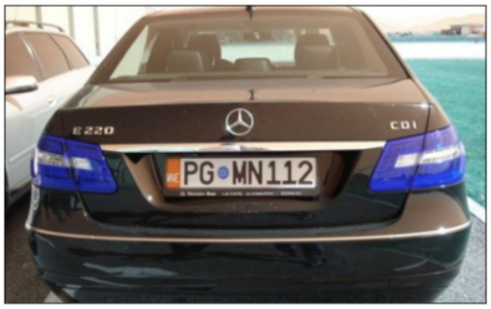


Figure 3.1 Original input image

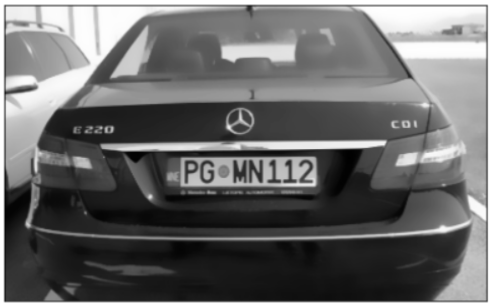


Figure 3.2 Gray image after bilateral filtering

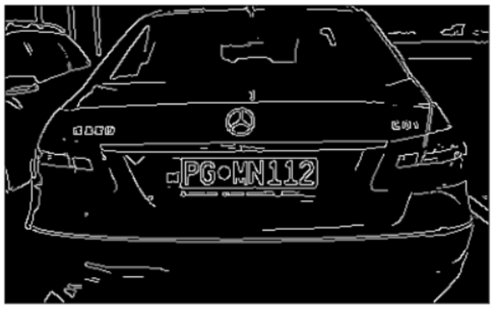


Figure 3.3 Canny Edge Detection

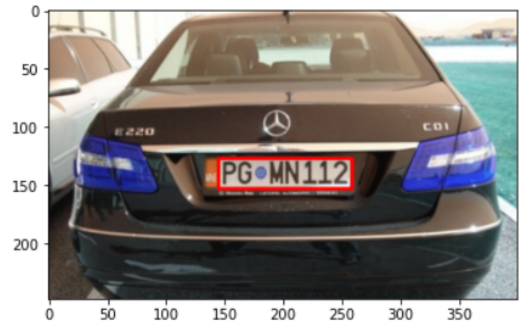


Figure 3.4 License Plate Detection

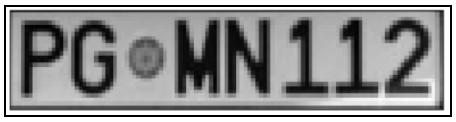


Figure 3.5 Cropped license plate

3.2 License Plate Detection

Canny edge detection with the findContours function was proposed to identify the license plate location. After the image preprocessing, edge detection is performed to localize the license plate, whereby it shows the edges of the image. The edges have mostly surrounded the characters on the license plate region. Canny edge detector in OpenCV will be used to perform edge detection as it can detect both horizontal and vertical edges while suppressing the remaining noise in the image. The sample output results are shown in Figure 3.3.

Next, to allocate the contours in the image, the findContours function of OpenCV was being used. The detected contours count the number of edges it bounds then sorted it in descending order, which is from big to small. When a maximum localized number of edges is found, that portion of the image will be detected as the license plate. This is because a license plate is a four-sided rectangle bounding. Figure 3.4 shows the license plate is detected surrounded by a red bounding box.

After that, masking is applied to the entire image excluding the region where the license plate is detected. This is because the detected license plate will be cropped out into a new image. The cropped license plate will then be saved into a folder in the directory and will be used for the following processes.

3.3 Character segmentation

Segmentation is to extract each of the characters on the license plate for identification. The method of combining threshold operations and the findContours function for character segmentation was proposed.

As for the first stage, we have converted the detected grayscale image into a binary image by thresholding The findContours function to obtain each of the individual characters from the license plate. Firstly, the height and width of the plate are divided and stored into a variable called “ratio”. We select a contour that has a height larger than 50% of the plate, and a bounding box will be drawn around the contours of the character for the plate’s ratios that are between 1 to 3.5. Next, each of the bounding boxes is separated and resized into a standard width and height which is the size of 30 x 60. Lastly, conversion to binary images is performed again by using the threshold method.

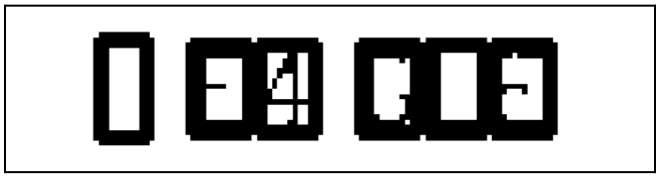


Figure 3.3 Bounding boxes drawn around detected characters.

Once, each of the characters in the license plate is segmented correctly, then it will proceed to the next stage which is character recognition.

3.4 Character Recognition

There are two supervised machine learning classification algorithms that will be implemented to recognize the license plate character after the segmentation process. We have implemented a feature extraction method where training the model begins with character segmentation by constructing a horizontal projection to the plate for detecting the character's position and space. Then the characters are segmented and built for the database. The characters were distributed into the sequence of A-Z, and 0-9 for further operations for our models.

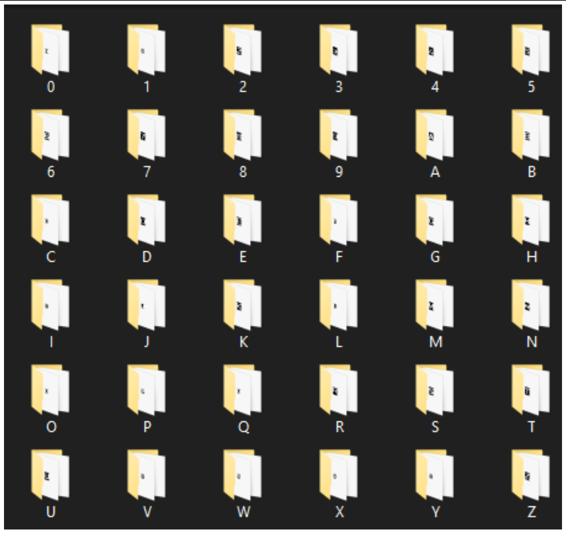


Figure 3.4.1 The distributed characters folder from A-Z, 0-9

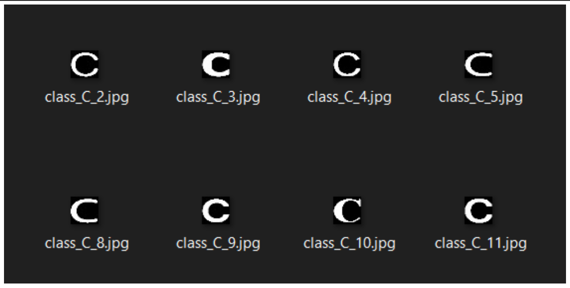


Figure 3.4.2 Sample of the segmented character "C"

3.4.1 OCR Algorithm- Python-tesseract

OCR is short for Optical Character Recognition. A two-dimensional image that contains text is reconstructed by OCR into a machine-readable text. OCR can be used on any type of image file, such as PDF document or even handwritten document. Python has an open-source OCR engine that supports a huge variety of languages. Currently, Tesseract has included a neural network subsystem that functions to recognize texts. After detecting text regions from an image with the OpenCV library, the ROI of text is then passed into Tesseract which will build an OpenCV OCR pipeline.

Tesseract needs to be installed and the directory of its installation needs to be stated on the .py file and set to path environment. This allows the terminal to pip install tesseract and runs it. The configuration of the directory plays an important part as any mistake in the steps stated above could result in the image files not being able to work along with tesseract.

There are several configuration options that are needed in order for the OCR model to work effectively. Page segmentation modes(psm), OCR engine modes(oem) and languages(l) need to be defined in the tesseract model. Psm needs to be set as it defines the way tesseract splits or segments images into several lines of words. Options that were chosen for the kaggle dataset was '--psm 7'. This takes an image and treats it as one single line. This configuration particularly worked well with the kaggle dataset as there were many unique fonts present. '-- psm 11' was most suitable with the Malaysia dataset as this configuration takes in sparse text and finds as much text in no particular order. Some new fonts that were detected by creating a bounding box and labeling them were fed into the training data to detect the different fonts. For both the datasets, the oem was set to 3. Oem helps to configure different engine modes and 3 is a default setting that uses what is available.

3.4.2 Artificial Neural Network

The second machine learning algorithm that is implemented is Artificial Neural Network (ANN). ANN was selected due to the ability to drive meaning from complicated or imprecise data, where it can be used to extract patterns and detect patterns that are too complex to be noticed by humans or any other machine learning algorithms.

Initially, the image data is first transformed to a gray level image then the image is binarized for contours extraction. The threshold is set by image size where it detects the x-axis of the image. If the width is greater than the threshold, the image will be cropped. The normalized character image size is 28 x 28 and it should match the template. Then the minimum bounding rectangle is calculated for further character segmentation. After finding the out bounding rectangle of the characters, it is easy to crop out the rectangle regions each with a character as a traditional segmentation method. The resized images were passed using an image data generator where random transformations on each training image are fed to the model. This will not only make the model robust but will also be able to save up on the overhead memory. Hence, the trained list of data will be used from feature extraction techniques to proceed further for training.

Next, the image will be fed into an ANN Model for training. During the training, parameters will be adjusted according to the number of iterations, batch size, hidden layer neurons, and other conditions. Once the training phase is done, the trained network is ready to use for test data. The testing input is fed into the input layer and the feed-forward network will generate results based on its knowledge from the trained network and the output will be stored into a '.csv' file format.

3.4.3 Random Forest Classification

The third machine learning technique for the evaluation comparison is Random Forest Classification. The reason for choosing Random Forest Classifier rather than Decision Tree is because of its flexibility resulting in a vast improvement in accuracy as Random Forest's trees do not correlate with each other.

Unlike OCR and ANN, Random Forest has a limitation on processing images, it could only process the images if the image has been converted to numeric data. Therefore, before performing the classifier, we employed the database which is the segmented characters that were distributed in sequence from A-Z, 0-9 to perform resizing action. After resizing the segmented characters into the size of 150 x 150, we then appended them into a NumPy array. To train the model to recognize the characters A-Z and 0-9, we created a list of categories from A-Z, 0-9, then appended them into another array. By performing these actions, the segmented characters will then be trained based on the list of categories. We split the data into 80% of the train set and 20% of the test set, and select 130 numbers of trees, "gini" for criterion, and the rest are following the default parameters of Random Forest Classifier.

We wish to save some time when running the models for the Kaggle and Malaysia datasets, we implemented joblib to save the classifier model. When running the saved classifier model, the output will be appended into a list and printed into a CSV file.

4. Experiment

This section displays and interprets the outputs by breaking down the performance of the algorithm with different image conditions.

4.1 Datasets

We will use 2 datasets for this research. One of the datasets is taken from Kaggle which contains 433 images of cars with license plates whereas the second dataset is the Malaysia license plate which contains 120 images of Malaysian cars taken by our group members. All these license plates vary from different angles, lighting conditions, and resolutions.

4.2 Quantitative evaluation

The score of Accuracy, Precision, Recall will be used to evaluate the performance of the three machine learning algorithms. By these scores, we will have a result on which machine learning algorithms performed the best in recognizing the character of license plates.

4.2.1 License Plate Detection

Accuracy is used to evaluate the performance of the license plate detection on the algorithm quantitatively. Equation 4.1.2 shows the formula to calculate accuracy for plate detection.

$$Accuracy = \frac{detected\ license\ plate}{total\ car\ images} \times 100\%$$

Equation 4.2.1: Accuracy formula for the plate detection

The evaluation method above calculates the accuracy by dividing the number of correctly detected license plates with the total number of car images, then multiplying by 100%. We will consider a successful detection of the license plates as long as the bounding box of the license plates is fully detected.

There were multiple duplicated images and unrelated car images in the Kaggle dataset. Therefore, among 433 car images, only 230 license plates were successfully detected. The accuracy obtained for the Kaggle dataset is 53.12%. On the other hand, for the Malaysia dataset, we were able to successfully detect 76 license plates from 120 car images. Thus, the Malaysia dataset obtained an accuracy of 63.33%. The accuracy performance is highly influenced by the quality of the images.

4.2.2 Character segmentation and recognition

Three different evaluation metrics were used to evaluate our work, as per license plate, and based on character segmentation with accuracy, recall, and precision.

Precision is used to measure how well detected regions agree with the corresponding ground truth. It was calculated using the image by taking the number of correctly predicted regions that had a corresponding ground truth and dividing it by the total number of correctly predicted regions and wrongly detected characters.

Secondly, recall measures how well the ground truth regions were detected. The calculator is performed by taking the number of correctly predicted regions that had a corresponding ground truth and dividing it by the total number of correctly predicted regions and missing detected characters.

Next, accuracy was calculated as the number of correct detections divided by the total number of predictions. The calculation for accuracy was based on the number of correctly classified regions divided by the total number of character segmentation.

$$Precision = \frac{Correct\ (TP)}{Correct\ (TP) + Incorrect\ (FP)} \times 100\%$$

Equation 4.2.2: Precision of the plate detection

$$Recall = \frac{Correct\ (TP)}{Correct\ (TP) + Missing\ (FN)} \times 100\%$$

Equation 4.2.3: Recall of the plate detection

Models	Accuracy (%)
OCR	38.26
ANN	13.91
RF	2.17

Table 4.2.1: Kaggle dataset results of the entire license plate

Models	Accuracy (%)
OCR	40.79
ANN	3.04
RF	2.63

Table 4.2.2: Malaysia dataset results of the entire license plate

Models	Accuracy (%)	Precision (%)	Recall (%)
OCR	52.8	53.23	67.6
ANN	33.51	33.98	48.04
RF	24.54	24.96	44.98

Table 4.2.3: Kaggle dataset results of the characters from license plate

Models	Accuracy (%)	Precision (%)	Recall (%)
OCR	52.01	51.98	57.89
ANN	6.92	7.36	8.83
RF	6.88	7.47	9.35

Table 4.2.4: Malaysia dataset results of the characters from license plate

The overall performance by checking the entire license plate, Tesseract's OCR prevailed to have the highest accuracy which is 38.26% in the Kaggle dataset. Which is swiftly followed by ANN with 13.19% and 2.17% from Random forest.

When it comes to comparing the accuracy, precision, and recall from the character's perspective, OCR produces an accuracy of 52.8%, a precision of 53.23%, and recall of 67.6%. ANN leads second with an accuracy of 33.51%, precision of 33.98%, and recall of 48.04%. Finally, random forest reigns last with an accuracy of 24.4%, precision of 24.96%, and recall of 44.98%.

From the Malaysian dataset, the OCR model has an overall accuracy of 40.79 % for the entire license plate. Although ANN is the leading second with an accuracy of 3.04% it is not far off from Random Forest that produced an accuracy of 2.63%.

OCR leads with an accuracy of 52.01%, precision of 51.98%, and recall of 57.89% on evaluating the Malaysian dataset by characters. ANN leads second with an accuracy of 6.29%, precision of 7.36%, and 8.83%. Random Forest follows closely to ANN with an accuracy of 6.88%, precision of 7.47%, and recall of 9.35%.

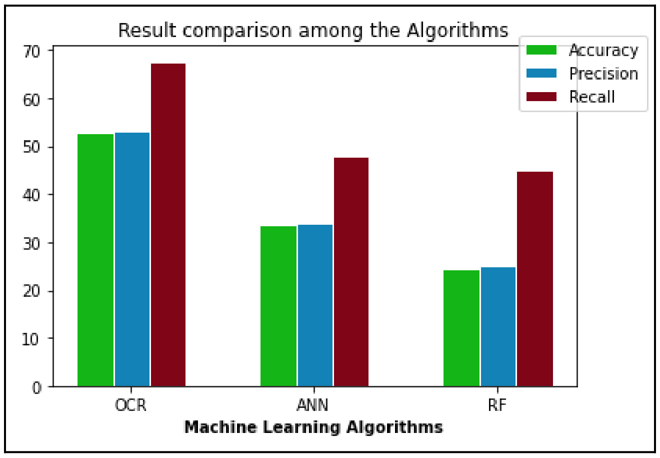


Figure 4.2.5: Kaggle dataset results of the characters from license plate

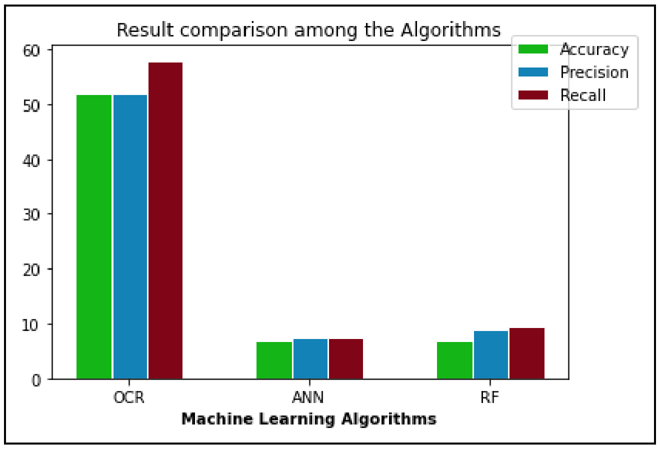


Figure 4.2.6: Malaysia dataset results of the characters from license plate

4.3 Qualitative evaluation

The qualitative analysis includes examining, comparing, contrasting, and interpreting patterns. Observation of the output images will be performed to determine whether the license plates were detected correctly and each of the characters was segmented accurately. Besides, observation is done to determine if lightning conditions and license plates' angles will affect the detection of license plates and recognizing the characters.

Moreover, investigating the reasons on the model resulted in good and bad output. Observation of the output will be performed to determine whether the license plates were in different conditions such as blurry, reflection, lighting conditions, and different angles on shooting the license plate that caused the generated output.

4.3.1 License Plate Detection

One major difference between the Kaggle dataset and the Malaysia dataset is the color of the license plate. Kaggle dataset contains car images worldwide, most of the license plate background color is white with black characters. However, the Malaysia dataset is on the opposite, the license plate characters are white with a black background.

We observed that when the car color is identical to the license plate's background color, the algorithm will fail to detect the license plate. The color of the license plate and the car color kind of blend together which makes it difficult to detect the actual position of the license plate. As we can see in Figure 4.3.1 the white car and the white license plate from the Kaggle dataset and the black car and black license plate from the Malaysia dataset. Both license plate color and the car color are almost alike, which makes it hard to detect the edges of the license plate. Hence, the license plate detection failed.

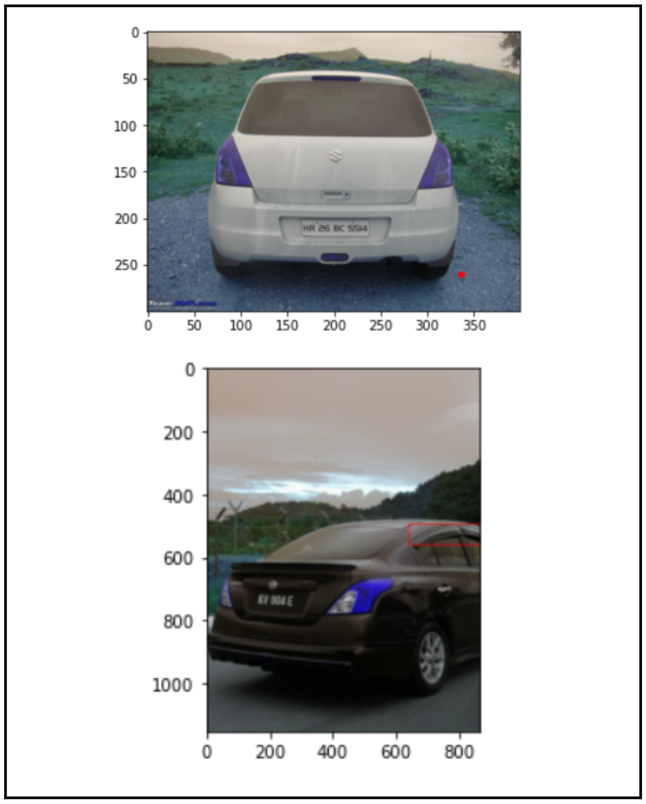


Figure 4.3.1: Background color affecting the detection

When the original image has good lighting conditions and the image is positioned facing the front with a straight angle, the algorithm is able to identify the location of the license plate accurately. Figure 4.3.2 shows the result when the image is having good lighting conditions with a straight angle positioned. The algorithm was able to detect the license plate perfectly.

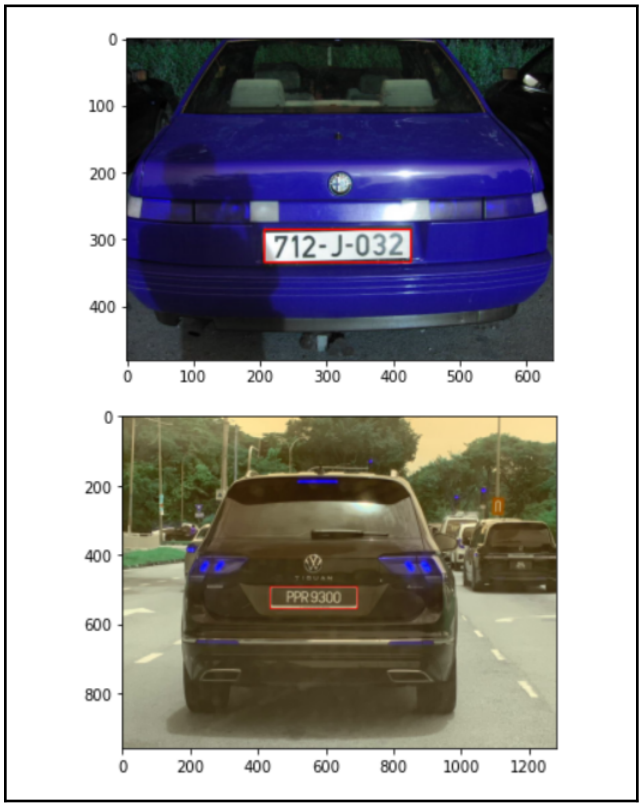
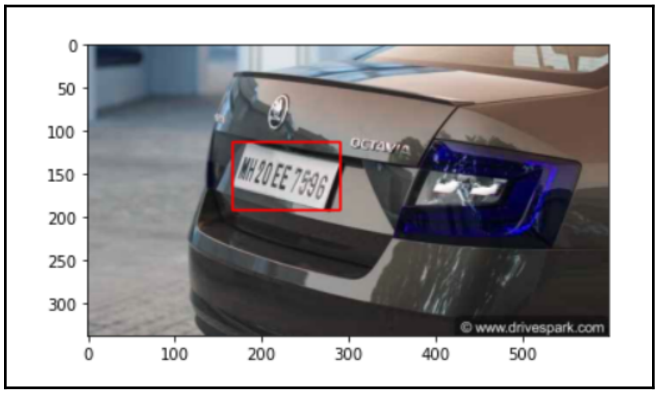


Figure 4.3.2: Results when the original image is in good lightning and perfect angle conditions

Although the original image is slightly angled, in some cases, the algorithm is still able to detect the license plate. Figure 4.3.3 shows the detected output when the image is at a slanted angle. When the original image of the license plate is not in a straight front angle, the location of the license plate can still be identified correctly, but the algorithm can still detect a small portion of the car part. However, we consider these results to be acceptable license plate detection and proceed with the character recognition.



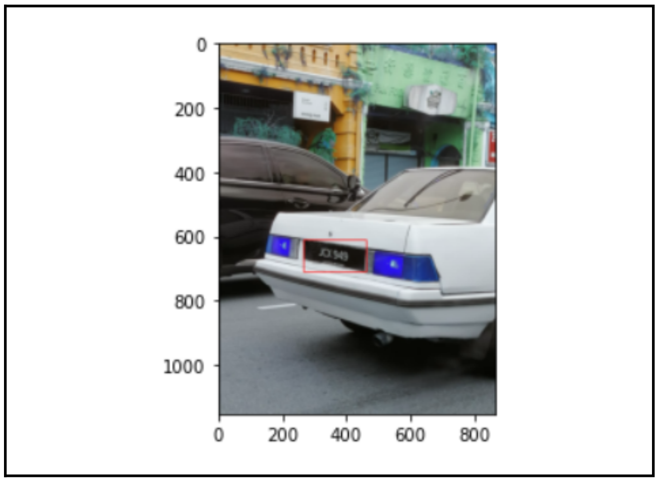


Figure 4.3.3: Results when the license plate is slanted

Besides, we also faced a few special cases whereby the detection of the license plates was abnormal. In Figure 4.3.4, the car image on the top, we can see that the algorithm detects a wide area of license plate even if the car lights were included. Additionally, in the car image on the bottom under Figure 4.3.4, only a portion of the license plate was detected. The algorithm missed out on the first character which is alphabet F on the license plate. For these few rare cases, we save the detected license plate images and proceed for character recognition to evaluate the performances.

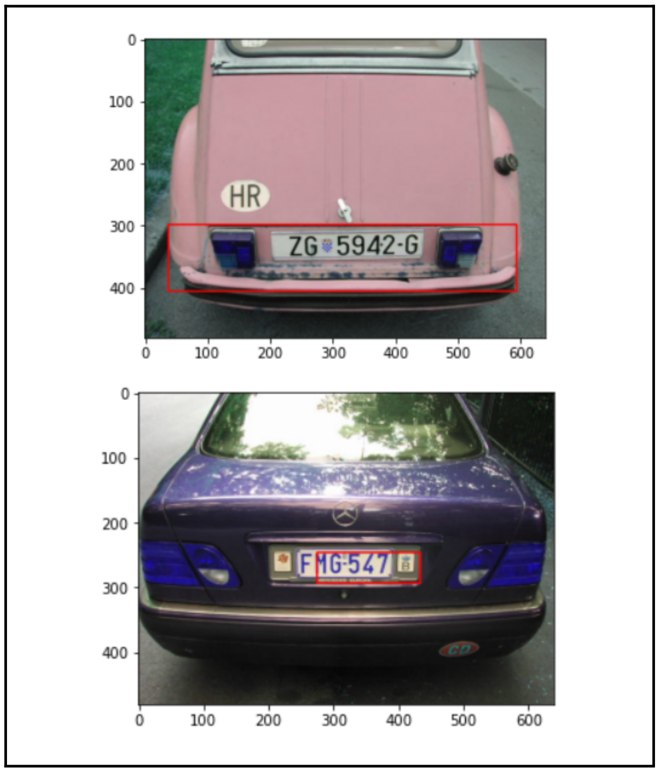


Figure 4.3.4: Results of special cases license plate detection

Moreover, when the license plate was reflected with extreme brightness or the condition of the car image was blurry and dark, the license plate failed to be detected. We can observe in Figure 4.3.5 both car images were not under good quality conditions, so the algorithm failed to identify the location of the license plate correctly. We can see that other unrelated parts were detected instead of the license plate.

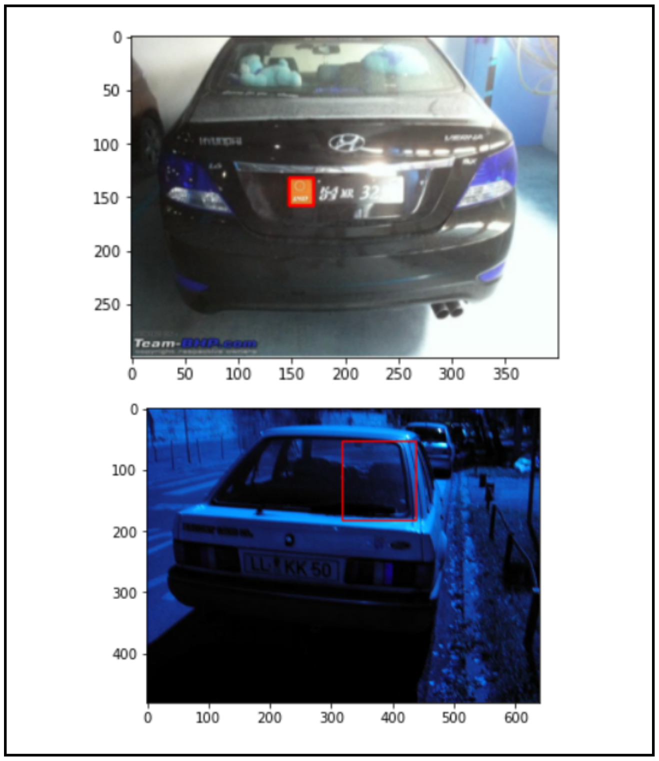


Figure 4.3.5: Results of failed detection of license plate

4.3.2 OCR

OCR essentially tries to detect the texts from the images. It also has its own set of limitations where it does not recognize any of the text at all. This is due to the model not detecting the font on the license plate. All these license plates do come with a standardized version of fonts but some have different angles and thicknesses which causes the OCR to confuse from one another. Some of the license plates have a very unique font that further confuses the OCR.

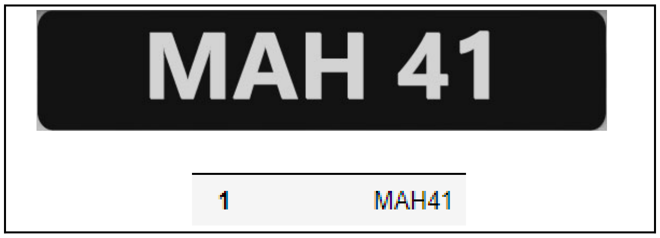


Figure 4.3.6 shows the correctly detected license plate and its text output that was detected by the OCR model.

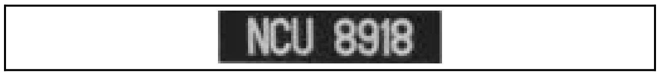


Figure 4.3.7 shows the license plate that was not detected at all.

From the preprocessing methods, the sizing of the cropped images also differs, which causes the OCR to misjudge the sizing. Images that are taken from different angles as well cause the OCR to not detect the right characters at all.

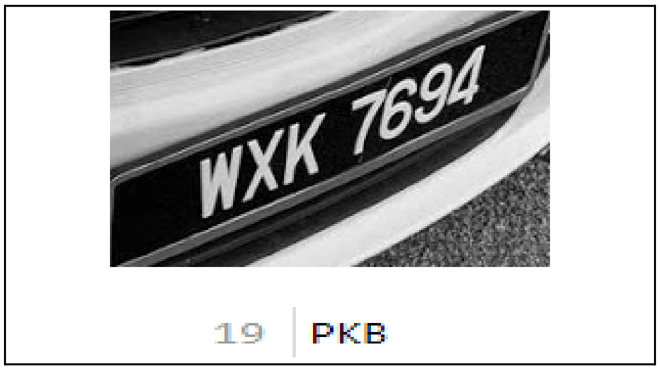


Figure 4.3.8: shows the image 'Cropped_19.png' that is tilted and the result that it has produced.

From the image above, it can be seen that the text from the license plate 'WXK 7694' is wrongly read at PKB. Some due to the size and blurriness of the image could not detect the right characters at all. During the experimentation process, applying filters for all the images caused the accuracy of detection to drop due to some images not needing them.

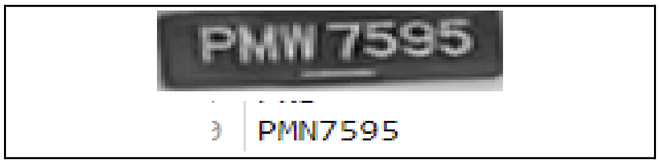


Figure 4.3.9: shows 'Cropped_20.png' a license plate and the text that was detected on it.

From Figure 4.3.9 the letter 'W' from the license plate ' PMW 7595' was wrongly detected at N, due to the blurriness. Some plates that have flaring or bright shadowing cause the image to detect special characters which later are removed through the regex.

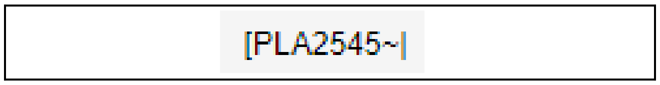


Figure 4.3.10: shows an output that contains special characters that were later removed through regex to further improve the accuracy.

4.3.3 Artificial Neural Networks

During the segmentation process, the model failed to detect some of the characters due to the resolution and dimension of the image as shown in Figure 4.3.3. The problem arises because the number plates are having a very small gap where there is not much spacing between the characters and that makes the segmentation fail. This problem can be resolved if the license plate has a larger gap among the characters.

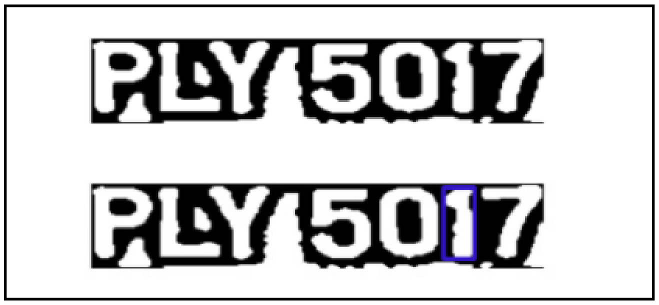


Figure 4.3.11: Failure to detect the characters (Cropped_70.png)

At some point, the contour function is unable to locate the correct segment of the license plate. As per Figure 4.3.11, the function detects an additional character that is not supposed to be part of the number plate and it recognizes different regions of the license plate as a random character. For instance, the model detects the ground truth 'E4GLE' as 'WE4GLE'.

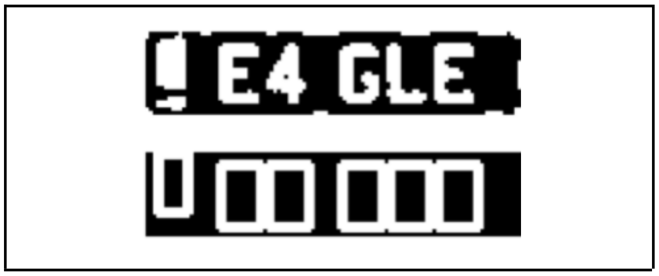


Figure 4.3.12: Additional characters detected (Cropped_1.png)

Moreover, Figure 4.3.12 shows how the function has some problems locating the license plate where the rectangle box is not drawn at the character 'M'.

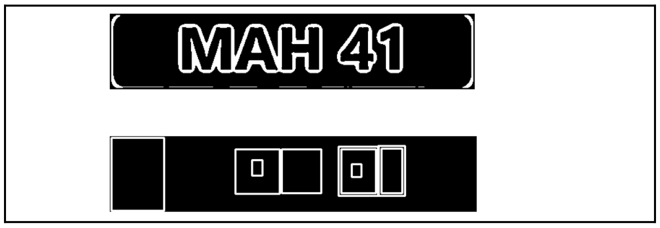


Figure 4.3.13: Failed to detect the character ('M': Cropped_1.png)

Not only that, the model is unable to detect certain characters that are similar to each other. For example, character '2' is detected as 'Z', '6' is detected as 'G', '1' is detected as 'l', etc. This is because the model is lacking training data where it needs more precise and accurate samples of characters during the training process to avoid the model from predicting the wrong output. Figure 4.3.14 shows a sample from the Malaysian Dataset, 'Cropped_3.png' where the alphabet '2' is wrongly predicted as 'Z'. In order to make the prediction better, the model should be able to differentiate between similar characters by adding different sizes of alphabets and based on different angles.

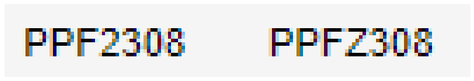


Figure 4.3.14: Sample of a wrongly predicted license plate.

Besides that, the model is unable to detect the edges of the license plate and it is having trouble locating images that are taken at a slanted angle. Figure 4.3.15 shows an example of images that are taken from a different angle and how the model fails to capture the rectangular box.

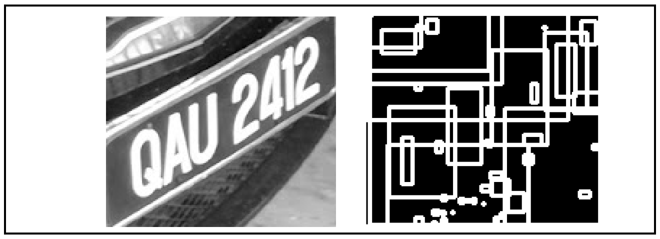


Figure 4.3.15: Model failed to recognize the location of ' 'cropped_19.png'

Not only that, there is also some failure in recognizing the number plate due to the fact that the license plate has reflecting lighting conditions which make the pictures have a blurry resolution. Figure 4.3.16 shows two different conditions where' reflection on license plate 'Cropped_18.png' and 'Cropped_19.png' has a darker shade of shadow appearing.

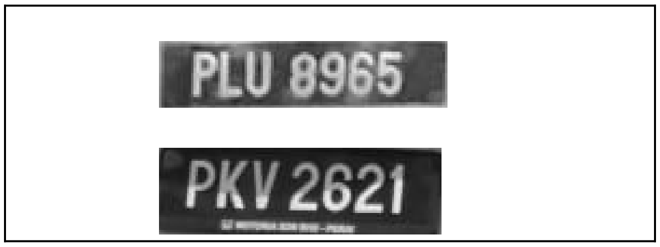


Figure 4.3.16: Reflections and shadows

Furthermore, the model is having a hard time recognizing the character in another font type especially in an Italic format where the minority of the vehicle is registered with the special words as shown in 'Cropped_120.png' and 'Cropped_50.png' in Figure 4.3.17. This is because the character template is predefined from the standard font type for the number plates.

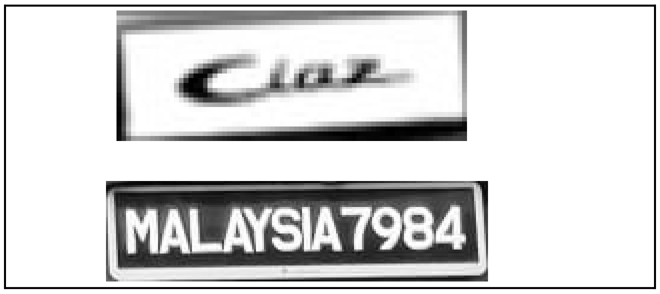


Figure 4.3.17: Italic words and predefined font

4.3.4 Random Forest Classifier

For the Kaggle dataset, due to the similarity between some characters such as 8 and B, 0 and U, the model wrongly predicted the plate character. For example, for Cropped_2.png and Cropped_8.png, the model wrongly predicted the '0' into 'U', but it predicted correctly for the rest of the characters. This could be further enhanced by feeding more character data to the model so that it could be trained to be more precise in differentiating the similarities for some characters.

Additionally, some characters of the license plate such as Cropped_18.png, Cropped_195.png, Cropped_203.png, Cropped_209.png, Cropped_212.png, and Cropped_219.png are not located at the same line but located at two different lines. For this situation, the model could not predict the characters at all.

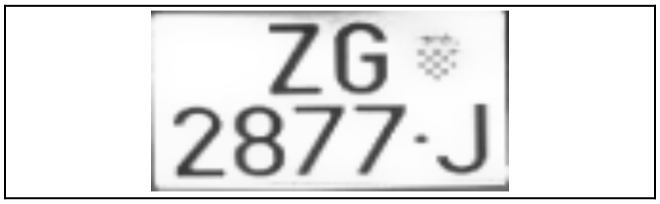


Figure 4.3.18: The characters that are located at two different lines. (Cropped_18.png)

The characters of the license plate named "Cropped_54.png" have the same font color as its background color, which means the character of the license plate is defined by its prominent shape. This license plate could not be detected at all as during the character segmentation and recognition operation, it does not segment and recognize any character of the license plate.

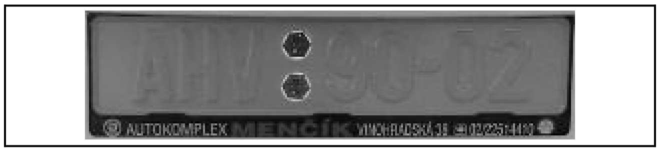


Figure 4.3.19: The font and background are having the same color. (Cropped_54.png)

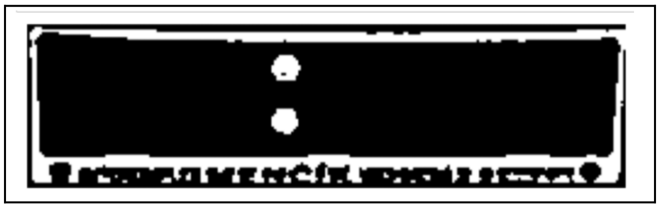


Figure 4.3.20: Cropped_54.png could not be recognized during the character segmentation and recognition operation.

Besides that, for Cropped_171.png, the license plate is different from others due to it having white font with a black background, unlike other license plates having black font with white background. Therefore, the threshold method should be using THRES_BINARY instead of THRES_BINARY_INV. However, all of the license plates in the dataset are running together at a time, it could not be specially using THRES_BINARY just for this license plate.

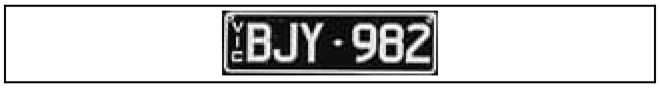


Figure 4.3.21: Cropped_171.png's font and background color are reversed from others.

Cropped_172.png and Cropped_173.png's shooting angle is from the side angle and up angle, but not from the front angle. Hence, because of the different angles, the model could not predict the license plate.

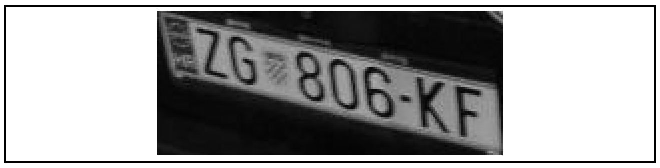


Figure 4.3.22: Cropped_172.png was shot from the side angle

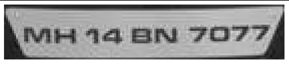


Figure 4.3.23: Cropped_173.png was shot from the up angle

For Cropped_184.png, this license plate image is a very small size with just 17 x 62 of size image after detection. The model was not able to predict the character from the license plate as it becomes very blurry after zooming in for segmentation and recognition.

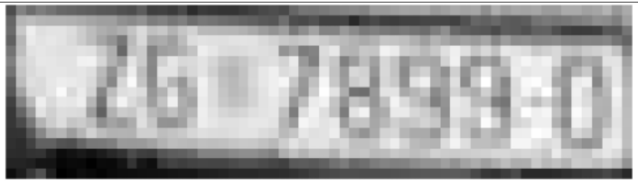


Figure 4.3.24: Cropped_184.png has become very blurry after zooming in

On the other hand, we have observed that the reasons for failure detection for the Malaysia dataset is also having the same limitation as to the Kaggle dataset such as the similarities of characters, different shooting angles, and the characters are in two different lines.

There is one limitation that the Malaysia dataset is having and the Kaggle dataset is not having is the reflection condition. Take Cropped_52.png for example, due to some reflection conditions, this license plate could not be recognized well.

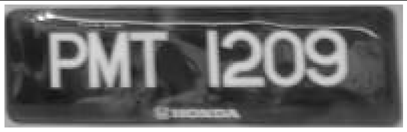


Figure 4.3.25: Cropped_52.png in Malaysia dataset is having some reflection

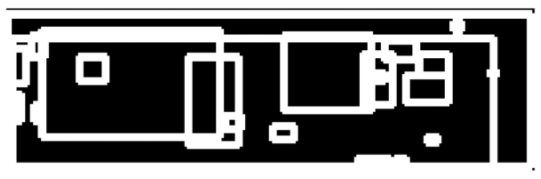


Figure 4.3.26: Cropped_52.png could not be recognized well

5. TASK DISTRIBUTION

Collecting data images, License plate detection, and Code compilation

- ❖ Gray image conversion, removal of noise, and contrast adjustment
- ❖ Thresholding: edge detection with Canny
- ❖ Finding Contours
  - ➢ (YEE WEN SAN)

Character segmentation & recognition and model evaluation

- ❖ OCR Algorithm- Python-pytesseract
  - ➢ (DHARISINI KANESAMOORTHY)
- ❖ Artificial Neural Networks
  - ➢ (PERIVITTA RAJENDRAN)
- ❖ Random Forest Classification
  - ➢ (LOU JIA YU)

Report Writing

- ❖ DHARISINI KANESAMOORTHY
- ❖ PERIVITTA RAJENDRAN
- ❖ LOU JIA YU
- ❖ YEE WEN SAN

6. CONCLUSION

In conclusion, this project aimed to assist humans in recognizing license plates automatically from images. Instead of manually checking the license plate one by one, our proposed method will be an alternative technique that effectively helps humans shorten the time used to recognize license plates. Automatic number plate recognition systems do not just record the numbers, but they do it in real-time, which gives a clear view of insights into monitoring the surveillance.

From the research that our group conducted, it is very noticeable that the OCR model performed much better with an overall accuracy of 38.26% on the Kaggle dataset, which is then followed by ANN with an accuracy of 13.91% and RF with 2.17%. On the Malaysian dataset, OCR performed better with 40.79% accuracy, which is followed by ANN with 3.04% and RF with 2.63% accuracy. Although the models did not perform as well as theorized, they can be improved on by in-depth preprocessing and different font training for the models. This project will be able to benefit security or traffic departments where it could be used to assist in inspections, forensics, investigations, and legal proceedings.

7. REFERENCES

[1] M. M. AlyanNezhadi, S. M. R. Hashemi and V. Abolghasemi, "License plate detection in complex scenes based on fusion of Gaussian filtering and Bayesian network," 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), 2017, pp. 0022-0026, doi: 10.1109/KBEI.2017.8324985.

[2] A. Menon and B. Omman, "Detection and Recognition of Multiple License Plate From Still Images," 2018 International Conference on Circuits and Systems in Digital

Enterprise Technology (ICCSDET), 2018, pp. 1-5, doi: 10.1109/ICCSDET.2018.8821138.

[3] Koval, Vasyl & Turchenko, Volodymyr & Kochan, V. & Sachenko, A. & Markowsky, George. (2003). Smart license plate recognition system based on image processing using neural networks 123 - 127,doi: 10.1109/IDAACS.2003.1249531.

[4] P. Faizal, S. Jaimini, R. Vivek and S. Ankit, " Recognition of Vehicle Number Plate Using Image Processing Technique" (2018), Medicaps Institute of Technology and Management Indore, 2018, doi: 10.18063/wct.v0i0.418

[5] R. V. Mario, M. Deekshith, P. Krishna Chaitanya, R. Angeline, "Object Detection: Automatic License Plate Detection using Deep Learning and OpenCV", International Journal of Engineering and Advanced Technology (IJEAT), 2019, ISSN: 2249 – 8958, Volume-9 Issue-1, 2019, doi: 10.35940/ijeat.A1842.109119

[6] Akhtar, Z., Ali, R. Automatic Number Plate Recognition Using Random Forest Classifier. SN COMPUT. SCI. 1, 120 (2020). https://doi.org/10.1007/s42979-020-00145-8

[7] Abdullah, Saman & Omar, Khairuddin & Sahran, Shahnorbanun & Khalid, Marzuki. (2009). License plate recognition based on Support Vector Machine. 78 - 82. 10.1109/ICEEI.2009.5254811.

[8] Shi X., Zhao W., Shen Y. (2005) Automatic License Plate Recognition System Based on Color Image Processing. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2005. ICCSA 2005. Lecture Notes in Computer Science, vol 3483. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11424925_121