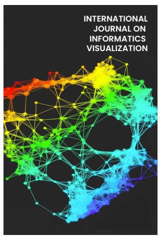
INTERNATIONAL JOURNAL
ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv



A Robust License Plate Detection System Using Smart Device

Muhammad Darwish Bin Mohamad Azhar, Kah Ong Michael Goh*, Law Check Yee, Tee Connie

Faculty of Information Science & Technology (FIST), Multimedia University, Jalan Ayer Keroh Lama, 75450 Melaka, Malaysia
Corresponding author: michael.goh@mmu.edu.my

***Abstract*— The license plate recognition (LPR) system is widely employed in various applications. However, most research studies have used a fixed camera rather than a moving one. This is because the location of the vehicle plate is nearly static and easily estimated, making the use of a static camera simple for locating and detecting the scanned license plate. Images obtained with a moving camera are highly complex due to frequent background changes. Additionally, a challenge with car plates in Malaysia is their non-standardized nature. Car owners are permitted to use any font type for their license plate number, rendering existing license plate recognition systems from other countries incapable of effectively detecting license plates on Malaysian car plates. A traditional LPR system typically requires a high-quality camera and a powerful computer for costly and bulky processing. Nowadays, many smartphones come equipped with powerful processors and cameras. Android smartphones include various libraries for modifying hardware configurations such as the camera. This paper presents a robust method for detecting Malaysia's license plate number using a convolutional neural network (CNN). The CNN model from the pre-training process is imported to the Android device and tested in real-time in an on-road driving environment, resulting in an average recognition rate of 89.37%. A comprehensive Character Recognition Analysis is also presented to demonstrate the accuracy of each character. However, there is still room for improvement in recognizing the character Q.**

***Keywords*— License Plate Recognition (LPR); Convolution Neural Network (CNN); Tesseract OCR**

Manuscript received 27 Oct. 2023; revised 7 Dec. 2023; accepted 20 Jan. 2024. Date of publication 31 May 2024.

I. INTRODUCTION

Smartphones are a necessity for everyone around the world nowadays. License Plate Recognition (LPR) systems are also necessary in today's environment. Different strategies are available for detection and recognition. However, detecting a plate with a smartphone is the most convenient one. It is a straightforward approach and can save time. Commercially accessible License Plate Recognition (LPR) systems are costly, heavily dependent on proprietary technology, and typically provided as part of comprehensive system packages [1]. Because of new advancements in smartphone technology, many smartphones come with powerful processors and cameras.

This project creates a smartphone-based license plate recognition and detection system. Using an Android smartphone's camera, this system lets users detect and recognize Malaysia's license plate number with different font-type characters in real-time. It can detect and identify with high accuracy and a fast response time. LPR systems integrated into mobile devices can be used by law enforcement officers on patrol. They can quickly capture license plate information of vehicles passing by and compare it against databases of stolen cars, wanted individuals, or vehicles with outstanding warrants. This enables law enforcement to identify and apprehend potential suspects or take appropriate action in real time.

II. MATERIALS AND METHODS

A. Software

TensorFlow, an open-source library, simplifies data collection, model training, prediction resolution, and future outcome improvement processes by integrating algorithms, machine learning, and deep learning models for large-scale machine learning and numerical computing [2]. Although TensorFlow is primarily created on the Python platform, TensorFlow models may be exported and used in any Android app.

TABLE I
LIST OF SOFTWARE REQUIREMENTS

Software	Purpose
TensorFlow	Construct CNN
Android Studio	Build and deploy Android applications.
Chipmunk	
Microsoft Office Excel	Recording results

CNN is pre-trained in TensorFlow. Android Studio is required as an IDE to develop and deploy the essential apps. Finally, all the test results are recorded. Table 1 summarizes the required software.

B. Hardware

This research aims to accomplish real-time LPR using Android. The smartphone is a high-end model with a robust CPU and a high-resolution camera. Huawei P20 Pro was used for the LPR process. The NN model is pre-trained on a laptop before being exported into an Android application. Tables 2 and 3 contain a list of all the necessary hardware.

TABLE II
ANDROID MOBILE PHONE SPECIFICATION DETAILS

Components	Specification
CPU	Octa-core (4x2.4 GHz Cortex-A73 & 4x1.8 GHz Cortex-A53) - HI Silicon Kirin 970
GPU	Mali-G72 MP12
RAM	6 GB
Internal Storage	128 GB
Network Data	GSM/ HSPA/LTE
OS	Android 10
Screen Size	6.1 inches
Screen Resolution	2240 x 1080
Camera Resolution	40MP, f/1.2, 27mm(wide)

TABLE III
LIST OF HARDWARE REQUIREMENTS

Hardware	Purpose
Huawei P20 Pro	LPR Camera
Lenovo Y520	Train CNN

C. Pre-Processing

Acquiring an image or video frame with a camera is the first step in an LPR system. In this study, the LPR process is carried out in real-time. The camera used is the smartphone camera itself. While the system runs, the footage is taken and transformed into video frames using the OpenCV function. The active development community for the OpenCV library continues to enhance its interface, promising an even more promising future for their integration [3]. The OpenCV function permits video frame conversion in the background of the process. A general flowchart of the pre-processing step is shown in Figure 1.

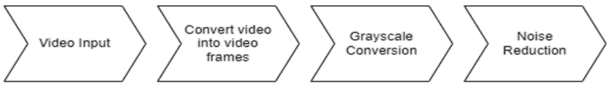


Fig. 1 Flowchart of Pre-Processing

Each converted frame will go through a few pre-processing procedures. Grayscale conversion is applied on the frame since grayscale simplifies the algorithm and minimizes processing needs compared to color photos, which require more color data [4]. Then, to keep edges while reducing noise, the median filter minimizes the noise in the frames. Median filtering is a non-linear method for smoothing signals, which relies on principles from statistical sorting theory to reduce noise [5] efficiently.

D. License Plate Localization

Based on the rectangular shape and border of the license plate, an LPR system may identify or extract a license plate image from a video frame in this second stage. In any LPR system, the significance of this phase lies in how accurately each segmented character affects character recognition accuracy [6]. The license plate has no fixed location because it might appear anywhere in the video frame. Several factors influence license plate localization, including illumination, image quality, and other factors [7]. The flowchart for the license plate localization stage is shown in Figure 2 below.

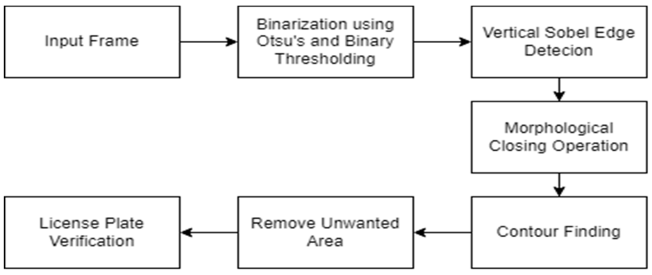

Input Frame
Binarization using Otsu's and Binary Thresholding
Vertical Sobel Edge Detecion
Morphological Closing Operation
Contour Finding
Remove Unwanted Area
License Plate Verification

Fig. 2 License Plate Localization Flowchart.

The proposed localization approach begins by converting the grayscale picture into a binary image. The Binarization distinguishes the target objects from the surrounding elements within the image [8]. The Otsu binarization approach and binary thresholding are applied in this study. Both thresholding methods are compared and discussed in this study. Binary thresholding sets a threshold value and compares each pixel to that threshold value. If the pixel value is less than or equal to the assigned threshold, then the pixel value is set to zero or the maximum value. Otsu's binarization is a straightforward and widely favored segmentation method focusing solely on differentiating foreground and background in an image by determining an optimal threshold through an exhaustive search to minimize variance [9].

The binary input frame is then applied to the edge-based method. Edge detection refers to recognizing sudden shifts in brightness or sharp changes in an image's intensity[10]. Edges can be identified through vertical edge detection, which isolates vertical lines, horizontal edge detection, which isolates horizontal lines, or a combined approach that captures an entire rectangular shape by simultaneously utilizing both methods [11]. Several edge-based operators, including the Sobel, Canny, and Prewit edge operators, are available. The Prewitt edge detector involves a higher level of computational complexity[12], and the Canny edge detection algorithm is complex and demands significant processing time [13]. Much like the Prewitt Operator, the Sobel Operator carries out a comparable computation but offers superior accuracy compared to the Prewitt Operator [14]. The Sobel edge detection method operates as a spatial filter using two distinct 3x3 convolutional kernels, one specialized in detecting vertical edges and the other in detecting horizontal edges [15]. The vertical Sobel operator retains this information because the alphabets and numerals on a license plate have more vertical edges than other letters.

Following that, the video frame is subjected to morphological action. Sequences of morphological operations are applied to eliminate extraneous outlines from the thresholder image [16]. Morphological opening involves the process of erosion followed by dilation [17], whereas Morphological closing combines dilation and erosion operations [18]. Dilation consists of increasing the number of pixels at the boundaries of objects in a digital image, whereas erosion is the reverse process [19]. The morphological closing operation subsequently bridges the spaces between the license plate characters and refines the outer edges [20].

Unwanted data within an image can hinder the precision of license plate recognition [21]. This includes trees, structures, and road signboards. To resolve this issue, contours from the video frame are recognized, and the area of each contour is analyzed. The evaluation of various factors, including minimum contour area, minimum contour width and height, and minimum and maximum possible aspect ratios, led to removing the majority of redundant contours [22]. This method eliminates vast items such as trees, buildings, road signboards, and minor objects such as dirt. The remaining contour is examined once again for license plate area verification.

After removing unwanted areas, the remaining contours are attached to a bounding box to verify the license plates. The height and width of the bounding box are used to analyze its features. The dimensions of the bounding boxes are determined based on the natural characteristics of license plate shapes [23]. The width of the bounding box for a license plate usually is more significant than the height. The bounding box's area is calculated. A huge bounding box area is removed. A non-license plate region is a bounding box that does not match this condition. The remaining components that meet the required criteria are selected for the license plate segmentation phase.

E. License Plate Segmentation

Following the detection of a license plate, conventional methods for license plate identification typically involve the segmentation of the plate characters to facilitate license plate recognition [24]. The segmentation task holds the utmost significance since it yields the outcome of the entire license plate analysis [25]. License plate segmentation isolates and extracts each character from a detected license plate image. The retrieved license plate character is then sent into NN as input for the character recognition procedure. Figure 3 depicts the flowchart of license plate segmentation employed in this study.

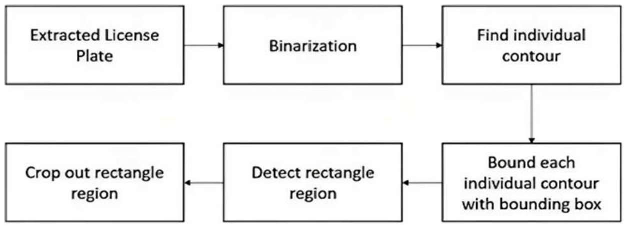


Fig. 3 License Plate Segmentation Flowchart.

The binarization approach is the first step in license plate segmentation on the retrieved license plate image. The Connected Component Analysis (CCA) approach is then used on the license plate image to determine individual connected contours. CCA is a widely used method for image processing involving an image analysis, wherein pixels are grouped into components based on their connectivity, indicating that all pixels within a connected component share the same pixel intensity level [26]. Following the separation of each character, the bounding box method is used to bind the digits and letters in the license plate image.

Finally, a rectangular region is identified, and all bounded characters are cropped out of the original image as a single input data. Cropping out each bounding character allows the character recognition algorithm to analyze each character. Before being sent for character identification into the NN, each cropped character is standardized to 28x28.

F. License Plate Recognition

The license plate character recognition stage is the final step in an LPR system. In this study, the NN technique is applied for character recognition. Multilayer perceptron, a neural network, is employed to recognize segmented characters [27]. The NN architecture for character recognition uses all segmented characters from the license plate segmentation procedure as input. CNN is the type of NN involved in this project. Convolutional Neural Networks (CNN) stands as a prominent deep learning algorithm capable of efficiently reducing the dimensions of large image datasets to smaller ones while preserving essential image features, making it one of the most extensively employed network architectures today [28]. The final experiment for CNN's LPR system accuracy is then recorded.

1) Convolutional Neural Network: To implement the CNN algorithm in Android devices, the Tesseract Optical Character Recognition (OCR) engine is used as it is prebuilt to handle Android mobile development. Tesseract OCR exhibits a relatively high accuracy when employed in License Plate Recognition (LPR) systems for moving vehicles [29]. However, because the Tesseract OCR engine is designed for text-based character recognition, the NN had to be retrained with the license plate character before it could be used in the LPR system [12]. The Chars74k dataset is used to retrain the network. The dataset contains 1000 images for each of the 62-character classes, for a total of 62000 images. TensorFlow is used as a framework to train the CNN model. After training, the resulting weight matrix model is exported to be used in the Android device. The advantage of this approach is that the NN architecture and weights matrix can be pre-trained on a high-performance desktop computer first, which reduces training time compared to training on the smartphone itself. The pre-training procedure produces the weight matrix and NN structure, which are then transferred into the Android device's NN for recognizing characters on license plates. The pre-training process's weight matrix and NN structure are transferred into the Android device's NN to recognize characters on license plates. However, the weights cannot be retrained. The CNN architecture is presented in Table 3.

TABLE IV
CNN ARCHITECTURE

Image input layer:	28x28x1
Convolutional layer #1:	8 3x3 filters
Convolutional layer #2:	16 3x3 filters
Convolutional layer #3:	32 3x3 filters
Activation Function:	ReLU
Pooling Layer #1:	2x2 max pooling with two strides
Pooling Layer #2:	2x2 max pooling with two strides

Fully connected layer:	62 neurons
Max Number of Epoch:	1000
Classification output:	Soft-max

2) Neural Network Training Datasets: Datasets, including characters and numeric digits, are required to train the NN for the LPR system. This scenario uses an open-source Chars74k [30] as the NN training and testing dataset. The dataset contains 74000 characters from natural images, handwritten characters, and computer fonts. All lowercase English characters 'a to 'z,' all uppercase English characters 'A' to 'Z,' and all digit number characters 0 to 9 are included in the collection's 62-character classes. The dataset images are in PNG format.

Only characters typeset using a computer font are used as the dataset for NN training and testing in this study. The dataset contains 62992 computer font images with 254 different fonts in four different styles: regular font, bold font, italic font, and bold italic font. However, not all images are used as training and testing datasets; images are first filtered, and only images that match license plate characters font are used.

III. Results and Discussion

A. Preliminary Testing and Results

Two sections in this preliminary testing discuss the results from each binarization method. Each section discussed the binarization experimental settings, which included the maximum distance between the LPR camera and license plate of each binarization and the binarization experimental results. The binarization experimental result is evaluated based on the number of boxes that bound the license plate and the number of boxes that appear that do not bind with the license plate. Boxes that are bound with the license plate mean it detects the license plate correctly, whereas boxes that are not bound with the license plate mean it wrongly detects the license plate with other things in the frame.

1) Distance of LPR Camera with License Plate: In the first experiment, the maximum distance between the camera and the license plate at which the LPR system can still recognize the entire string of license plates is determined. The LPR camera won't be able to detect and identify the license plate accurately when it is moving farther away from it. Therefore, this experiment aims to determine the optimal distance between the target license plate and the LPR camera of each binarization method to recognize license plates.

Only one car is needed to carry out this experiment. The illustration of the experiment setup is shown in Figure 4. Throughout the test, the vehicle with the target license plate stays in place and motionless. The person with the LPR camera is initially 4ft behind the car, the optimal distance for the license plate to fit in the camera's frame. The camera is then focused on the license plate of the car. Following is the process of the experiment:

- A person with an LPR camera walks backward slowly until the LPR system cannot recognize the car's complete license plate.
- The person with the LPR camera stops when the LPR system is incorrect or cannot recognize one or more characters on the license plate, and the maximum distance is recorded.
- Steps 1 and 2 are repeated three times, and the results are recorded in Table V.
- Repeat all the steps with another binarization method.

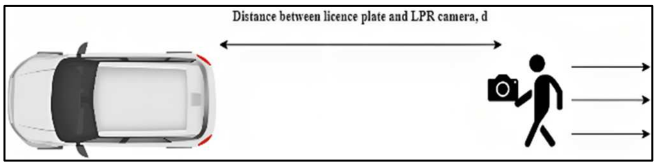


Fig. 4 A Person slowly walks backward while holding an LPR camera pointed at a parked car license plate.

TABLE V
COMPARISON OF MAXIMUM DISTANCES BETWEEN LPR CAMERA AND LICENSE PLATE

(A) OTSU'S THRESHOLDING

Binarization Type	**Otsu's Thresholding**									
Trial	1	2	3	4	5	6	7	8	9	10
Distance, d (ft)	9.5	10	9.5	9	9	9.5	10	9.5	9	9.5
Mode of Distance (ft)	9.5									

(B) BINARY THRESHOLDING

Binarization Type	**Binary Thresholding**									
Trial	1	2	3	4	5	6	7	8	9	10
Distance, d (ft)	10	10	9	10.5	10	10	10.5	10	10	10
Mode of Distance (ft)	9.5									

Based on the findings, when using Otsu's thresholding, the LPR system can accurately identify a license plate with a mode distance of 9.5 ft from the target license plate. Meanwhile, for binary thresholding, it can accurately identify a license plate with a mode distance of 10 ft from the target license plate. Therefore, it is best to keep the camera and the target license plate within a 9 to 10 feet range while testing.

2) Binarization Experimental Results: An experiment running LPR with Otsu's thresholding and binary thresholding was conducted. The experiment is carried out in good weather to ensure the best result. The process is observed. The range between the car in front is always within 9 to 10 feet during the testing. The process of the experiment is as follows:

- A person with an LPR camera is within 9 to 10 feet of the targeted license plate.
- The person with the LPR camera screen captures the process of analyzing several wrongly detected boxes.
- Repeat the first and second steps with another binarization method.

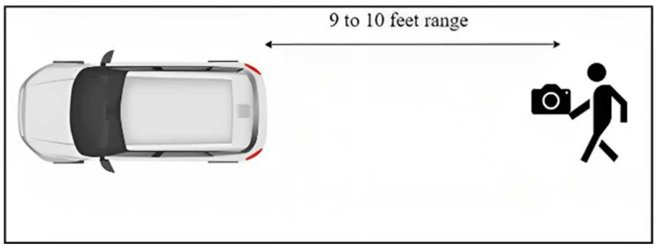


Fig. 5 Person with LPR camera pointed at a parked car license plate in 9 to 10 feet range.

Based on the testing results, both thresholding methods can localize the license plate within a 9- to 10-foot range. However, when using Otsu's thresholding, the LPR system frequently detects things other than the license plate. Meanwhile, binary thresholding does not show any wrongly detected boxes. Figures 6 and 7 show the result of the binarization experiment.

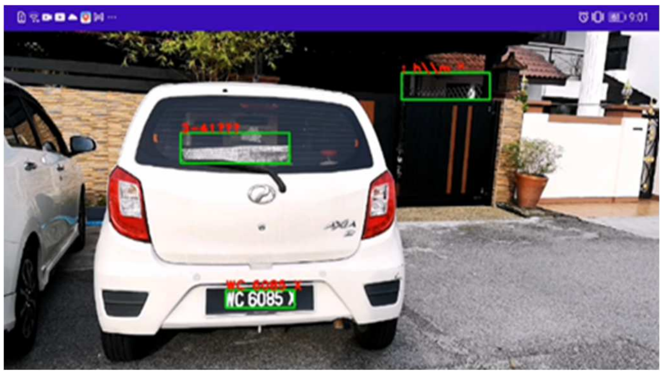


Fig. 6 Result of Otsu's Thresholding

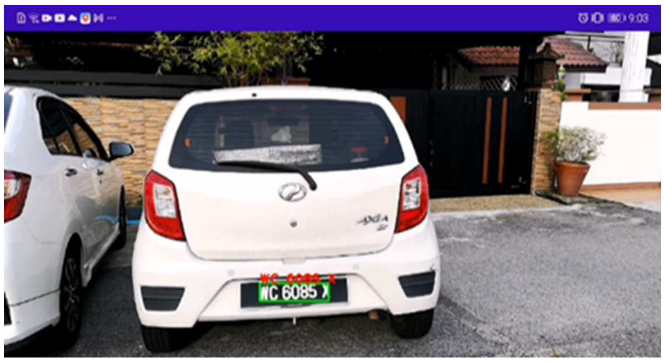


Fig. 7 Result of Binary Thresholding.

Besides that, the processing speed of both thresholding methods was observed. The processing speed when using Otsu's thresholding was plodding. The frame rate of the LPR system was constantly dropping as it frequently detected things other than the license plate, making it very slow. However, the system maintains a decent frame rate with binary thresholding, and the processing speed is better than Otsu's thresholding. Therefore, with the result of this preliminary experiment, binary thresholding is applied and used to test LPR accuracy as it gives good results overall.

B. System Testing

1) Experimental Results: The experiment is carried out in good weather to ensure the best result. The process of LPR is recorded, and the outcome is observed and manually recorded. The range between the cars in front must always be within a 9 to 10 feet range during the testing. The LPR rate is calculated using the following formulas:

$$LPRR\ (\%) = \left(\frac{W}{X}\right) \times 100 \qquad (1)$$

$$ALPRR\ (\%) = \left(\frac{S}{T}\right) \times 100 \qquad (2)$$

In equation 1, LPRR (%) is the LPR rate, W is the number of characters recognized on the license plate, and X is the total number of characters on the license plate. In equation 2, ALPRR (%) is the average license plate recognition rate, S is the sum of license plate recognition rates, and T is the total number of license plates.

TABLE VI
LPR EXPERIMENTAL RESULT

Total Number of License Plates:	87
Number of License Plate Fully Recognised:	40
Number of License Plate Partially Recognised:	47
Average Recognition Rate:	89.37%

Based on the result, the LPR can achieve an average LPR rate of 89.37%. There have been several times when the LPR system has been unable to fully recognize the license plate string but can still recognize some parts of it. Several factors might cause the license plate not to be recognized fully, such as the camera losing focus and moving too quickly, causing some of the license plate characters to appear unclear in the video frame.

2) Character Recognition Analysis: The experimental results are further analyzed to determine the recognition accuracy rate of each character. This analysis only considers the characters of fully and partially recognized license plates. The total number of each character from the detected license plate is accumulated. The number of characters correctly recognized is recorded. In partially recognized license plates, a few times the LPR camera misses out on some characters or falsely recognizes the character as other letters or digits; the characters it is falsely recognized as are recorded as well. Finally, the accuracy rate of each character is calculated using the following formula. CRAR (%) is the character recognition accuracy rate, Y is the number of characters correctly recognized, and Z is the total number of characters.

$$CRAR\ (\%) = \frac{Y}{Z} \qquad (3)$$

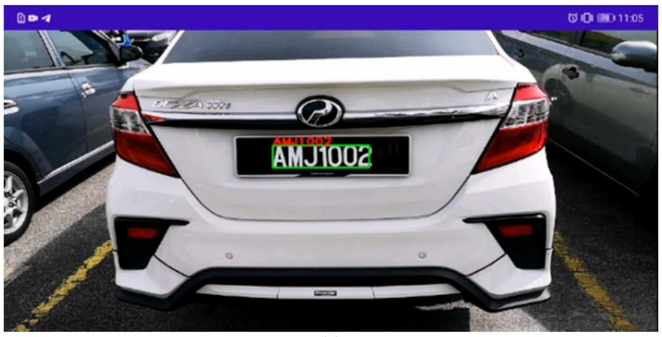
TABLE VII
SUMMARY OF CNN CHARACTER RECOGNITION RATES

Characters	**Total Number of Characters**	**Number of Character Correctly Recognized**	**Number of Character Falsely Recognized**	**List of Character Falsely Recognized As (If Any)**	**Accuracy Rate (%)**
0	35	28	7	5, 6, C	80%
1	30	30	0	N/A	100%
2	33	33	0	N/A	100%
3	34	27	7	2, 5	79.41%
4	39	39	0	N/A	100%
5	28	28	0	N/A	100%
6	22	10	12	5,3,8,0	45.45%
7	30	27	3	1	90%
8	47	42	5	5,3,6	89.36%

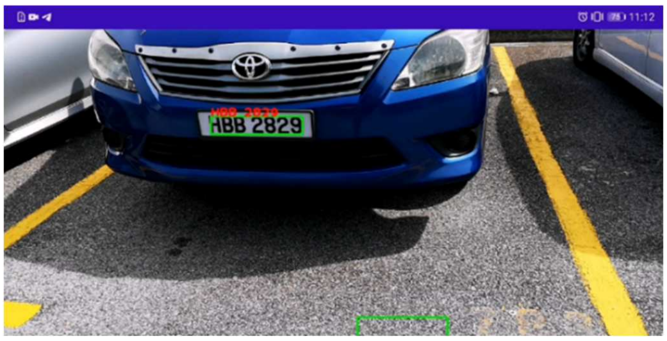
Characters	Total Number of Characters	Number of Character Correctly Recognized	Number of Character Falsely Recognized	List of Character Falsely Recognized As (If Any)	Accuracy Rate (%)
9	36	31	5	5,3,	86.11%
A	15	15	0	N/A	100%
B	15	14	1	E	93.33%
C	14	13	1	O	92.86%
D	13	12	1	U	92.31%
E	4	4	0	N/A	100%
F	3	3	0	N/A	100%
G	7	5	2	E, 5	71.43%
H	6	6	0	N/A	100%
I	0	0	0	N/A	N/A
J	17	17	0	N/A	100%
K	11	11	0	N/A	100%
L	1	1	0	N/A	100%
M	9	9	0	N/A	100%
N	10	10	0	N/A	100%
O	0	0	0	N/A	N/A
P	13	13	0	N/A	100%
Q	8	3	5	O, U, G	37.5%
R	6	3	3	E, B	50%
S	8	8	0	N/A	100%
T	3	3	0	N/A	100%
U	6	6	0	N/A	100%
V	30	30	0	N/A	100%
W	37	28	9	N, M	75,68%
X	7	7	0	N/A	100%
Y	8	8	0	N/A	100%
Z	0	0	0	N/A	N/A

Table VII shows the summary of the character recognition accuracy rate. Based on the table, characters of ‘1’, ‘2’, ‘4’, ‘5’, ‘A’, ‘E’, ‘F’, ‘H’, ‘J’, ‘K’, ‘L’, ‘M’, ‘N’, ‘P’, ‘S’, ‘T’, ‘U’, ‘V’, ‘X’, and ‘Y’ can achieve a 100% recognition success rate. There are some characters where the LPR was missing detected in the license plate, which resulted in not being able to achieve a 100% recognition accuracy rate. These include characters of ‘0’, ‘3’, ‘6’, ‘7’, ‘8’, ‘9’, ‘B’, ‘C’, ‘D’, ‘G’, ‘Q’, ‘R’, and ‘W’. Characters ‘I’, ‘O’, and ‘Z’ data are unavailable for this testing to calculate their recognition accuracy rate.

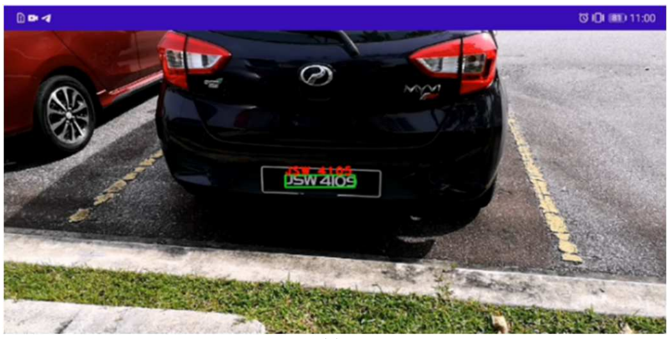
In some cases, LPR mistakenly detects characters as other letters or digits due to their similarity. For example, character ‘Q’ is mistakenly recognized as character ‘O’, ‘U’, ‘G’, and character ‘R’ is mistakenly recognized as character ‘E’ and ‘B’ and vice versa. The result also shows that character ‘Q’ has the lowest character recognition accuracy rate, 37.5%.



(a)



(b)



(c)

Fig. 8 Example of LP recognition process. (a) is a general number plate. (b) is a taxi’s number plate. (c) is a license plate with a different font style.

IV. Conclusion

This research work studied the implementation of the LPR system in a real-time environment with a neural network on a mobile device. Several experiments have already been conducted and discussed to examine specific parameters of the LPR system. The LPR system is tested in good weather conditions only when enough bright light enters the camera lens. Future work can be conducted by testing the LPR system in different weather situations, such as rain, cloudy weather, or night-light conditions. This will increase the difficulty in detecting and recognizing the license plate.

This work performs the license plate detection step by implementing image processing steps such as Sobel edge detection and morphological-based method. Future works can be considered to apply car object detection and license plate object detection methods using deep learning. This method requires a lot of car and license plate image datasets for the deep learning models. This approach will provide a much more accurate license plate localization process, thus eliminating non-license plate objects detected in the system. Further investigation of the implementation of license plate object detection with deep learning in real-time using the Android LPR system might yield different results.

Acknowledgments

This work was supported by the TM R&D Fund (Project no. RDTC/221054 and SAP ID: MMUE/220023) and MMU IR Fund (Project ID MMUI/220041).

References

[1] M. C. Rademeyer, A. Barnard, and M. J. Booysen, “Optoelectronic and Environmental Factors Affecting the Accuracy of Crowd-Sourced Vehicle-Mounted License Plate Recognition,” *IEEE Open Journal of Intelligent Transportation Systems*, vol. 1, pp. 15–28, 2020, doi:10.1109/OJITS.2020.2991402.

[2] K. M. N, S. S. S, S. E, S. D, and D. R, “A Novel Deep Learning Approach for Number Plate Detection,” in *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*, IEEE, Feb. 2023, pp. 281–286. doi:10.1109/ICCMC56507.2023.10084177.

[3] C. Duan and S. Luo, “Design of License Plate Recognition System Based on OpenCV,” in *2022 2nd International Conference on Networking Systems of AI (INSAI)*, IEEE, Oct. 2022, pp. 16–21. doi:10.1109/INSAI56792.2022.00013.

[4] C. Kanan and G. W. Cottrell, “Color-to-grayscale: Does the method matter in image recognition?” *PLoS One*, vol. 7, no. 1, Jan. 2012, doi:10.1371/journal.pone.0029740.

[5] M. Li, Z. Miao, J. Wang, S. Wang, and Y. Zhang, “A Method of License Plate Recognition Based on BP Neural Network with Median Filtering,” *J Phys Conf Ser*, vol. 1229, no. 1, p. 012033, May 2019, doi: 10.1088/1742-6596/1229/1/012033.

[6] C. Liew, C. Kim On, R. Alfred, T. Tse Guan, and P. Anthony, “Real-time mobile based license plate recognition system with neural networks,” *J Phys Conf Ser*, vol. 1502, no. 1, p. 012032, Mar. 2020, doi: 10.1088/1742-6596/1502/1/012032.

[7] Q. Zhu, Y. Liu, Z. Zhao, and X. Ma, “Research on License Plate Location Algorithm Based on YOLOv5,” *J Phys Conf Ser*, vol. 2278, no. 1, p. 012040, May 2022, doi: 10.1088/1742-6596/2278/1/012040.

[8] L. P. Saxena, “Niblack’s binarization method and its modifications to real-time applications: a review,” *Artif Intell Rev*, vol. 51, no. 4, pp. 673–705, Apr. 2019, doi: 10.1007/s10462-017-9574-2.

[9] S. Khairnar, S. D. Thepade, and S. Gite, “Effect of image binarization thresholds on breast cancer identification in mammography images using OTSU, Niblack, Burnsen, Thepade’s SBTC,” *Intelligent Systems with Applications*, vol. 10–11, p. 200046, Jul. 2021, doi:10.1016/j.iswa.2021.200046.

[10] S. Elmi and Z. Elmi, “A robust edge detection technique based on Matching Pursuit algorithm for natural and medical images,” *Biomedical Engineering Advances*, vol. 4, p. 100052, Dec. 2022, doi:10.1016/j.bea.2022.100052.

[11] Lubna, N. Mufti, and S. A. A. Shah, “Automatic Number Plate Recognition:A Detailed Survey of Relevant Algorithms,” *Sensors*, vol. 21, no. 9, p. 3028, Apr. 2021, doi: 10.3390/s21093028.

[12] C. Liew, “Moving Camera Automatic Number Plate Recognition Using Neural Network in Android Platform,” 2019.

[13] L. Guo and S. Wu, “FPGA Implementation of a Real-Time Edge Detection System Based on an Improved Canny Algorithm,” *Applied Sciences*, vol. 13, no. 2, p. 870, Jan. 2023, doi: 10.3390/app13020870.

[14] X. Zhang, “Optimize Vehicle License Plate Recognition System by Image Recognition,” in *2023 IEEE 5th International Conference on Power, Intelligent Computing and Systems (ICPICS)*, IEEE, Jul. 2023, pp. 1025–1029. doi: 10.1109/ICPICS58376.2023.10235383.

[15] P. Mukhija and P. Priyanka, “Comparison of Different Edge Detection Techniques Used in License Plate Localization,” in *2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT)*, IEEE, Jul. 2022, pp. 400–403. doi: 10.1109/CCiCT56684.2022.00077.

[16] A. A. Ahmed and S. Ahmed, “A Real-Time Car Towing Management System Using ML-Powered Automatic Number Plate Recognition,” *Algorithms*, vol. 14, no. 11, p. 317, Oct. 2021, doi:10.3390/a14110317.

[17] H. Kaur, J. Virmani, Kriti, and S. Thakur, “A genetic algorithm-based metaheuristic approach to customizing a computer-aided classification system for enhanced screen film mammograms,” in *U-Healthcare Monitoring Systems*, Elsevier, 2019, pp. 217–259. doi: 10.1016/B978-0-12-815370-3.00010-4.

[18] M. Mie Aung, “Study for License Plate Detection,” *International Journal of Image, Graphics and Signal Processing*, vol. 11, no. 12, pp. 39–46, Dec. 2019, doi: 10.5815/ijigsp.2019.12.05.

[19] Y. P. Pasrun, M. Muchtar, A. N. Basyarah, and Noorhasanah, “Indonesian License Plate Detection Using Morphological Operation,” *IOP Conf Ser Mater Sci Eng*, vol. 797, no. 1, p. 012037, Mar. 2020, doi: 10.1088/1757-899X/797/1/012037.

[20] C. Liew, C. Kim On, R. Alfred, T. Tse Guan, and P. Anthony, “Real-time mobile based license plate recognition system with neural networks,” *J Phys Conf Ser*, vol. 1502, no. 1, p. 012032, Mar. 2020, doi: 10.1088/1742-6596/1502/1/012032.

[21] K. Khan, A. Imran, H. Z. U. Rehman, A. Fazil, M. Zakwan, and Z. Mahmood, “Performance enhancement method for multiple license plate recognition in challenging environments,” *EURASIP J Image Video Process*, vol. 2021, no. 1, p. 30, Dec. 2021, doi:10.1186/s13640-021-00572-4.

[22] R. K. Varma P, S. Ganta, H. K. B, and P. Svsrk, “A Novel Method for Indian Vehicle Registration Number Plate Detection and Recognition using Image Processing Techniques,” *Procedia Comput Sci*, vol. 167, pp. 2623–2633, 2020, doi: 10.1016/j.procs.2020.03.324.

[23] S. M. Silva and C. R. Jung, “A Flexible Approach for Automatic License Plate Recognition in Unconstrained Scenarios,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5693–5703, Jun. 2022, doi: 10.1109/TITS.2021.3055946.

[24] F. Sultan, K. Khan, Y. A. Shah, M. Shahzad, U. Khan, and Z. Mahmood, “Towards Automatic License Plate Recognition in Challenging Conditions,” *Applied Sciences*, vol. 13, no. 6, p. 3956, Mar. 2023, doi: 10.3390/app13063956.

[25] N.-A.- Alam, M. Ahsan, Md. A. Based, and J. Haider, “Intelligent System for Vehicles Number Plate Detection and Recognition Using Convolutional Neural Networks,” *Technologies (Basel)*, vol. 9, no. 1, p. 9, Jan. 2021, doi: 10.3390/technologies9010009.

[26] A. Pattanaik and R. C. Balabantaray, “Enhancement of license plate recognition performance using Xception with Mish activation function,” *Multimed Tools Appl*, vol. 82, no. 11, pp. 16793–16815, May 2023, doi: 10.1007/s11042-022-13922-9.

[27] A. Ammar, A. Koubaa, W. Boulila, B. Benjdira, and Y. Alhabashi, “A Multi-Stage Deep-Learning-Based Vehicle and License Plate Recognition System with Real-Time Edge Inference,” *Sensors*, vol. 23, no. 4, p. 2120, Feb. 2023, doi: 10.3390/s23042120.

[28] S. Zhang, X. Lu, and Z. Lu, “Improved CNN-based CatBoost model for license plate remote sensing image classification,” *Signal Processing*, vol. 213, p. 109196, Dec. 2023, doi: 10.1016/j.sigpro.2023.109196.

[29] A. S. Rawat, H. Devrani, A. Yaduvanshi, M. Bohra, I. Kumar, and T. Singh, “Surveillance System using Moving Vehicle Number Plate Recognition,” in *2023 2nd International Conference on Edge Computing and Applications (ICECAA)*, IEEE, Jul. 2023, pp. 940–945. doi: 10.1109/ICECAA58104.2023.10212426.

[30] T. E. de Campos, “Character Recognition in Natural Images,” 2009. [Online]. Available: http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/