International Journal on Advanced Science Engineering Information Technology

Vol.14 (2024) No. 4
ISSN: 2088-5334

Vehicle License Plate Detection and Recognition using OpenCV and Tesseract OCR

Kalaphath Kounlaxay [a], Yeo Chan Yoon [b], Soo Kyun Kim [b,*]

[a] *Department of Computer Engineering, Souphanouvong University, Louangprabang City, Laos*
[b] *Department of Artificial Intelligence, Jeju National University, Jeju City, Republic of Korea*
[c] *Department of Computer Engineering, Jeju National University, Jeju City, Republic of Korea*
*Corresponding author: *kimsk@jejunu.ac.kr*

***Abstract*—License plate recognition (LPR) is essential as the number of vehicles increases and the human ability to accomplish this task is limited. If human labor is used to manage these, it will take a lot of time and energy and cause a discrepancy. License Plate Recognition (LPR) is an advanced technology that leverages optical character recognition (OCR) and various image processing methods to read vehicle license plates automatically. Typically, an LPR system comprises two primary components: detecting vehicles and their license plates and recognizing the alphanumeric characters displayed on those plates. This study explores the use of OpenCV for license plate detection and Tesseract OCR for character recognition. In this research, the dataset for training and testing the system included 100 license plates evenly split between plates featuring English and Lao characters. The Lao license plates presented unique complexities due to their specific characteristics. The experimental setup involved processing images of license plates taken from multiple angles. The system's performance was evaluated based on the speed and accuracy of line and character recognition. For English character plates, the recognition process took 0.12 seconds with an accuracy of 98.8%. In contrast, the Lao character plates required 0.24 seconds, achieving an accuracy rate of 89.42%.**

***Keywords*—Computer vision; license plate; ROI; OpenCV; tesseract OCR.**

Manuscript received 20 Dec. 2023; revised 23 Feb. 2024; accepted 8 Apr. 2024. Date of publication 31 Aug. 2024.

I. INTRODUCTION

LPR technology continues to evolve with advancements in computer vision, machine learning, and deep learning, improving accuracy and reliability in recognizing license plates under various conditions. The number of vehicles on the roads is increasing rapidly, and most of the time, verifying the identity of these vehicles is very important for the authorization of traffic regulations and supervision of parking lots [1]. It is difficult to check this colossal number of moving vehicles physically. LPR has become an important topic due to the increasing interest of modern technological society in powerful, intelligent surveillance and security systems [2]. Thus, LPR systems can be utilized as part of a traffic monitoring system with traffic lights to identify vehicles violating traffic rules or detect prohibited vehicles [3]. However, identifying vehicle license plates has always been challenging for various reasons, such as changes in brightness, irregular vehicle shadow license plate character types, multiple styles, and environmental color effects [4]. LPR technology is structured around three primary steps: LP detection, character segmentation, and character recognition, which are essential for detecting and identifying specific or all objects in an image [5]. In computer vision [6], systems have been designed to perform tasks that traditionally require constant human supervision and decision-making. These tasks typically involve detecting, identifying, and tracking objects of interest [7]. Although object detection [8] and object recognition [9] are related techniques, they differ in how they are implemented. Object detection focuses on locating instances of objects within images. With the integration of deep learning, object detection extends into a form of object recognition that identifies an object and recognizes it within its visual context, enabling the identification of multiple objects within the same frame.

This study is organized into two main segments: the detection and recognition of license plates with annotations. The initial phase involves pinpointing the location of the license plate based on its distinctive features. The subsequent phase entails segmenting characters on the plate to distinguish

individual alphanumeric characters. These characters are then processed through an optical character recognition (OCR) engine, which accurately identifies and annotates each character on the license plate.

A. Background of Object Detection and Recognition

This section outlines the foundational concepts necessary for this study, focusing on object detection and recognition. These topics encompass the core ideas, underlying principles, and techniques pertinent to their implementation.

1) Object Detection: This is a crucial area within computer vision and image processing that involves identifying specific categories of semantic objects—such as people, buildings, or vehicles—in digital images and videos. Therefore, object detection is a critical method of machine learning and deep learning [11]. Additionally, Object detection in images has been identified as a crucial area of computer vision image processing research [12]. The purpose is to instruct machines to understand (perceive) the content of images, similar to what humans do, as shown in Figure. 1.

Fig. 1 Differences between Classification, Location and Detection

The above figures demonstrate the results of image classification, object localization, and object detection. In classifying images, the system takes an image as input and outputs a classification label for that image along with some units of measurement such as probability, loss, accuracy, etc. For example, a picture of a car can be classified using the class label "car", or an image of a person can be classified using the class label "person" with some probability. A detector outcome commonly consists of a list of bounding boxes, confidence levels, and classes [13]. The object detection algorithms calculate object locations by finding whether an object is present in an image and displaying it with a bounding box [14]. It takes an image as input and outputs the position of the bounding box in terms of position, height, and width. Finally, the object detection algorithm combines image classification and object localization [15]. The image will be input, and then one or more bounding boxes will be created with a class label attached to each bounding box [16]. These algorithms can handle multiclass classification and localization tasks, including dealing with objects with multiple events.

Various algorithms and models can be used for object detection, and one popular approach is convolutional neural networks (CNNs). The basic idea is to divide the image into a grid and predict the presence or absence of an object in each grid cell and the bounding box coordinates [17]. Here is a simplified representation of the formulas used in object detection:

- Grid Division: Divide the input image into a grid. Each grid cell is responsible for detecting objects present in that region.
- Bounding Box Prediction: Predict the bounding box coordinates for each grid cell. A bounding box typically represents the coordinates $(x_{top\text{-}left}, y_{top\text{-}left})$ of the top-left corner and the dimensions (width, height). The bounding box coordinates for the button-right corner $(x_{bottom\text{-}right}, y_{bottom\text{-}right})$ can be calculated as follows:

$$x_{bottom-right} = x_{top-left} + width \quad (1)$$

$$y_{bottom-right} = y_{top-left} + height \quad (2)$$

- Object Class Prediction: For each grid cell, the probability of the presence of different classes of objects is predicted. This is usually performed using SoftMax activation to obtain class probabilities.
- Combine Predictions: Combine the bounding box coordinates and class probabilities to form the final predictions for each object detected in the image.
- Non-maximum suppression: To remove duplicate or highly overlapping bounding box predictions, apply non-maximum suppression which this step ensures that only the most confident and non-overlapping bounding boxes are retained.
- Intersection over Union (IoU): The IoU between two bounding boxes is calculated as the ratio of the intersection and combined areas. The intersection area is the overlapping area between two bounding boxes, and the union area is the sum of the total areas of both bounding boxes.

$$IoU = \frac{Area\ of\ intersection}{Area\ of\ Union} \quad (3)$$

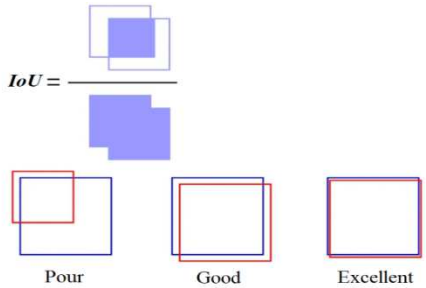


Fig. 2 A diagram representing the intersection over union

- Object Confidence Score: where *P(Object)* is the probability of an object being present in the grid cell and *P(Class)* is the probability of belonging to a particular class.

$$Confidence\ Score = P(Object) \times P(Class) \quad (4)$$

Object detection models, especially those based on deep learning, are trained using labeled datasets, where each object in an image is annotated with its class label and bounding box coordinates. Popular architectures for object detection include the Single Shot Multibox Detector (SSD), You Only Look Once (YOLO), and Region-Based CNNs (R-CNNs). The specific formulas and details may vary based on the chosen model architecture.

2) Object Recognition: Object recognition is a technique in computer vision that identifies various objects within images or videos. It is a crucial application of deep learning and machine learning technologies [18]. People, objects, scenes, and visual details can be instantly observed when humans view photos or watch videos. The goal is to teach computers to do something natural for humans: to make sense of what is in an image [19]. Object recognition using deep learning CNN is one of the most popular object methods [20]. CNN is used to detect objects in an environment [21], and it is widely used, and most state-of-the-art neural networks use this method to perform a variety of tasks. About object recognition, such as image classification, a CNN network takes images as input and outputs probabilities of different classes [22]. If there is an object in the picture, the output may be high; otherwise, the output probabilities of the remaining categories will be very small or very low, and model training is performed when the error rate decreases. Then, the trained model tests some sample images [23]. Compared with machine learning, the advantage of deep learning is that there is no need to extract features from the data.

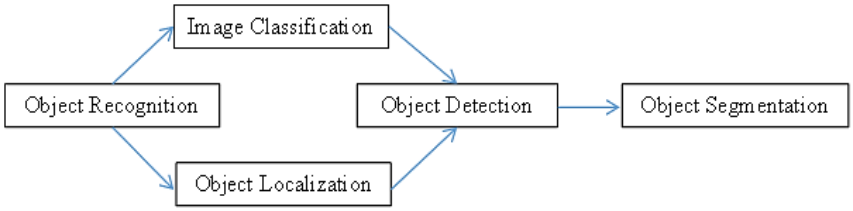


Fig. 3 Overview of tasks related to object recognition

Object recognition involves several interconnected tasks, each contributing to identifying and understanding objects inside the images or video frames. Each of these tasks plays a crucial role in the overall object recognition pipeline, and advancements in technology, especially in deep learning, continue to refine and enhance the capabilities of object recognition systems. Object recognition involves identifying and classifying objects within an image. Thus, CNNs are commonly used for object recognition tasks [24]. Below is a simplified formula for object recognition using a deep learning model:

- Input image: Let I be the input image
- Preprocessing: Preprocess the image, including normalization and resizing, if necessary.
- CNN: A pre-trained CNN model is used to train a model for object recognition. Popular architectures include VGG, ResNet, Inception, and MobileNet.
- Forward Pass: A forward pass-through CNN is performed to obtain predictions for the input image.
- Class Probabilities: Obtain class probabilities using a SoftMax activation function.

$$P(class_i|I) = \frac{e^{z_i}}{\Sigma_{j=1}^{N} e^{z_j}} \quad (5)$$

Here, z_i is the raw output (logit) for *class i*, and N is the total number of classes.

- Class Prediction: The class with the highest probability is the predicted class for the object.

$$Predicted\ Class = \arg max\ P(class_i|I) \quad (6)$$

- Probability Threshold: Optionally, a probability threshold is applied to filter out low-confidence predictions.

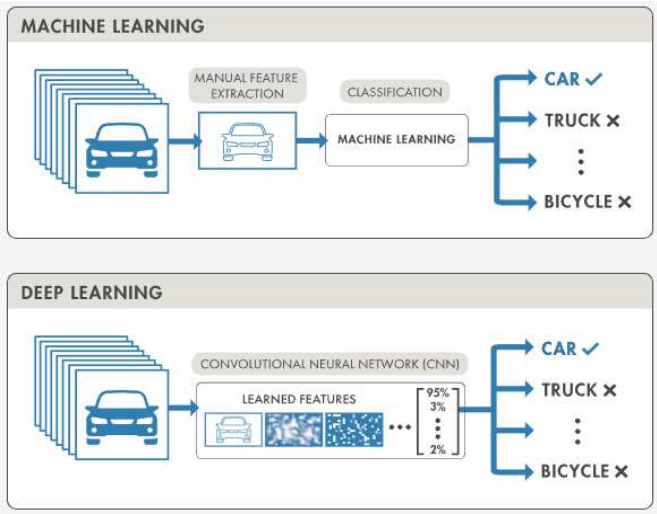


Fig. 4 Machine learning and deep learning techniques for object recognition

The output was generated by the last layer, and the fully connected nature of the CNN model is a single layer of labels. Therefore, the CNN method will not perform if multiple class labels are present in the image [25]. To limit the presence of objects in the bounding box, we need to try another approach other than exporting class labels. However, it also exports the position of the bounding box.

3) OpenCV: OpenCV is an open-source library that serves both computer vision and machine learning functions. The term computer vision, abbreviated as "CV" in OpenCV, refers to a field of research that enables computers to understand digital image content, such as photos and movies. Understanding the content of an image is the goal of computer vision [26]. Thus, OpenCV and Python are increasingly being applied to the shape and color detection of images [27]. A description of the image is used, and the image may belong to

the object. Text descriptions or three-dimensional (3D) models are extracted from images [28].

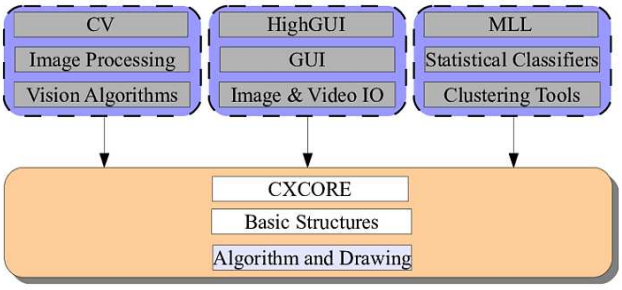


Fig. 5 OpenCV software architecture

In addition to stereopsis, computer vision may be used for various purposes. Stereo vision and motion tracking, pattern detection, augmented reality, scene reconstruction, and human-machine interaction are all included in this technology category [29]. For example, computer vision can help cars by enabling them to recognize objects, such as pedestrians, traffic signs, and traffic lights, along the road and then respond appropriately. Computer vision is crucial in various traffic monitoring, management, and safety applications. Computer vision is commonly applied in traffic-related scenarios.

Fig. 6 Object detection and recognition on traffic roads

The image above is a real traffic road in Vientiane, Laos, that demonstrates computer vision through the use of OpenCV for object classification, detection, and recognition.

4) Grayscale Conversion: In OpenCV, converting an image to grayscale involves combining the red, green, and blue (RGB) channels of the image through a weighted average. A frequently used technique for this process is known as the luminosity method. The luminosity method considers the human eye's sensitivity to different colors and assigns weights accordingly. Grayscale images are mainly used for shape features, edge detection, round objects, corner points, etc. [30]. The function for converting an RGB image to grayscale using the luminosity method is:

$$\text{Gray} = 0.299 \times \text{Red} + 0.587 \times \text{Green} + 0.114 \times \text{Blue} \quad (7)$$

This function uses weights of 0.299, 0.587, and 0.114 for the RGB channels. These weights are based on the standard coefficients for the luminosity method. Alternatively, we can use the average method, which takes the simple average of the RGB values:

$$\text{Gray} = \frac{\text{Red}+\text{Green}+\text{Blue}}{3} \quad (8)$$

We can also use other methods, such as weight loss or custom weighting schemes, based on your specific needs. An image with two colors, black and white, is called a grayscale image. Black is considered the least intense color. In comparison, white is the highest color. According to the intensity contrast evaluation, computers render each pixel in a grayscale image based on light intensity. Many essential image characteristics may need to be restored during the conversion process, including sharpness, shading, contrast, and color structure. Furthermore, current methods consume large amounts of computing time and memory [31].

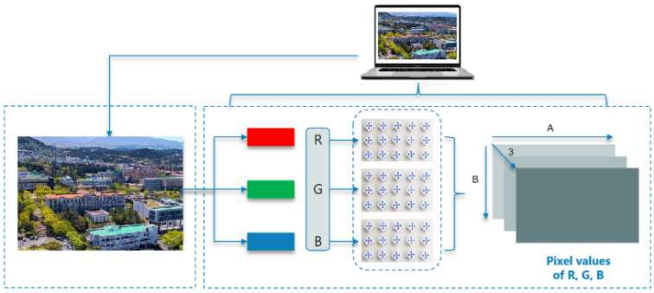


Fig. 7 Computer reading image

The computer reads each image in pixel values 0 to 255. For each color image, there are three main channels: RGB. It works very simply, a matrix is formed for each primary color, and these matrices are combined to provide pixel values for the individual RGB colors. Each element of the matrix offers data about the brightness intensity of the pixel, as shown in Figure 7, in which image size can be calculated by the formula: $B \times A \times 3$.

5) Image Thresholding: Thresholding is a widely utilized technique in image processing and computer vision that segments an image into different regions according to the intensity or color of the pixels. The core concept of thresholding involves converting pixel values to binary (typically 0 or 1), depending on whether they fall above or below a predefined threshold value [32].

Let us denote the input image as *I(x,y)*, where *x* and *y* are pixel coordinates. The global thresholding process involves assigning binary values to pixels based on a threshold value *T*. Typically, pixels with intensity values greater than *T* are assigned one value, such as 255 for white, and pixels with intensity values less than or equal to *T* are assigned another value, such as 0 for black. The thresholding function *T(x,y)* can be expressed as:

$$\begin{cases} 1 & if\ I(x,y) > T \\ \square & \\ 0 & if\ I(x,y) \le T \end{cases} \quad (9)$$

where *I(x,y)* represents the pixel's intensity at coordinate *(x,y)* in the input image and *T* is the global threshold value. This process results in a binary image, where pixels are assigned one of two values based on a comparison with the threshold. Additionally, these binary results can be helpful for tasks such

as object detection, edge detection, or simplifying further image analysis [33]. In practice, image-processing libraries such as OpenCV provide functions for performing thresholding without the need for explicit mathematical expressions. For instance, the *cv2.threshold()* function in OpenCV can apply global thresholding to an image.

6) Contour Detection: Contour detection is a computer vision technique used to identify and track the boundaries of objects in images. A contour is a curve along a boundary connecting all consecutive points of the same color or intensity. Contour detection is widely used in image processing and computer vision applications for object recognition, shape analysis, and image segmentation [34]. Additionally, the contour variable contains a list of contours; each contour is represented as an array of points. While the process involves multiple steps and functions, contour detection is a complex topic, and the specific methods used can vary depending on the application and the characteristics of the images being processed. In mathematical terms, a contour is often represented as a set of *(x, y)* coordinates that define the boundary of an object.

- Preprocessing: Often, contour detection begins with preprocessing steps, such as converting the image to grayscale to simplify the data and enhance edges. Other techniques, such as blurring or thresholding, may be applied to improve the quality of the image for contour extraction.
- Edge Detection: Edge detection algorithms such as Sobel, Canny, or Laplacian filters can be applied to identify areas of rapid intensity change in an image. These edges are crucial for contour detection.
- Contour Finding: The contours are then identified using algorithms like the one provided by OpenCV's '*findContours*' function. This function detects contours in a binary image, where the contours are represented as a list of points or a hierarchy of contours.
- Hierarchy and Approximation: Contour information may include hierarchical relationships between contours, helping to distinguish between inner and outer contours. Additionally, contours can be approximated to reduce the number of points and simplify the representation.
- Visualization: Finally, the contours can be visualized by drawing them onto the original image. This step helps in understanding and analyzing the detected objects.

Contour detection is fundamental in various computer vision applications, such as object recognition, gesture recognition, image segmentation, and robotics [35]. It allows computers to understand the shapes and boundaries of objects within an image, facilitating subsequent analysis and decision-making processes.

7) Region of Interest (ROI): An ROI is a specific area within an image or dataset relevant to a particular task or analysis. Identifying an ROI allows for more focused and efficient processing, reducing the computational load and improving the accuracy of the analysis. This process is commonly used in computer vision, machine learning, and image analysis applications [36]. There are various types of ROIs, and the choice of ROI depends on the specific requirements of the task. In image processing and computer vision in image analysis, an ROI might be defined by specifying the coordinates, drawing a bounding box, or segmenting an image to focus on a particular object or area of interest [37]. This approach is common in object detection, image recognition, and tracking tasks. Object tracking and detection are among the most common uses of ROIs in computer vision. Object detection refers to detecting the presence of an object of interest in an image or video frame. In contrast, object tracking involves finding a specific object of interest in a sequence of images or video frames.

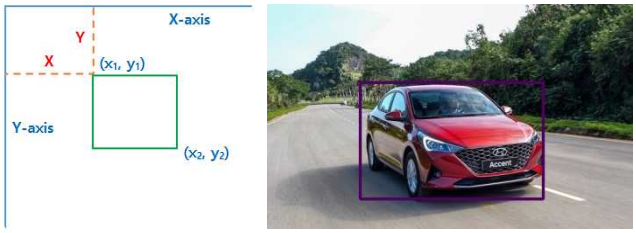


Fig. 8 ROI in object detection

8) Tesseract OCR: OCR is a key technology for converting typed or handwritten text into machine-readable text. It allows automatic data processing and analysis. However, OCRs for languages with complex text present unique challenges due to the complex nature of the characters [38]. Furthermore, Tesseract OCR, initially developed by HP Labs in 1985, was later released as an open-source project by Google in 2005 [39]. It has been under development since 2006. A combination of the OCR engine and improved text-matching algorithms are used to implement the system [40]. Tesseract is compatible with Unicode (UTF-8) and can recognize over 100 languages from the start, making it suitable for developing language scanning applications. The newer releases, like Tesseract 4, incorporate an OCR engine based on a Neural Net (LSTM) primarily focusing on online recognition [41]. Additionally, it supports the traditional Tesseract OCR engine, which identifies characters based on their patterns [42].

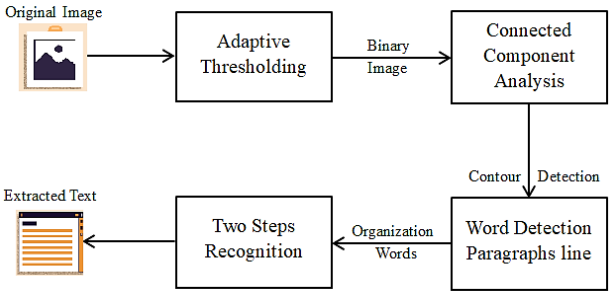


Fig. 9 Tesseract OCR Architecture

Tesseract OCR is commonly employed to transform documents, including scanned papers, PDF files, or images taken with a digital camera, into editable and searchable data. A comparison between Tesseract OCR's performance with English language support and its capabilities with multi-language support reveals significant differences, as detailed in Table 1.

TABLE I
COMPARISON OF TESSERACT OCR WITH ENGLISH LANGUAGE SUPPORT VS. MULTI-LANGUAGE SUPPORT

	English Only	Multi-language
Language Support	Tesseract OCR with English language support is explicitly optimized for English text recognition. It may perform better on English-language documents than the multi-language version when processing English text.	The multi-language version of Tesseract OCR supports a broader range of languages, making it more versatile for documents that contain text in multiple languages. It can handle languages other than English, including those with different character sets.
Accuracy	Tesseract OCR with English language support may achieve higher accuracy when processing English text because it is trained and optimized for the nuances and patterns specific to the English language.	While the multi-language version can handle a variety of languages, its accuracy might be slightly lower for English than that of the English-dedicated version.
Resource Usage	The English-only version of Tesseract OCR may be lighter in terms of resource usage since it only needs to load and process the data relevant to English language patterns.	The multi-language version may require more resources (memory and processing power) due to the broader language coverage and additional language models.
Training and Customization	If we have specific requirements for English text recognition and want to fine-tune the OCR engine for our specific use case, using the English-only version may offer more straightforward training and customization options.	The multi-language version provides flexibility for working with documents in various languages, but training and customization might be more complex due to the diverse language models involved.

Therefore, our use case depends on the choice between Tesseract OCR with English language support and the multi-language version. If documents primarily contain English text, using the English-only version might yield slightly better results and be more resource-efficient.

II. MATERIAL AND METHOD

The license plate, a critical vehicle component, can be detected and recognized in images or video streams using LPR technology that combines OpenCV and Tesseract OCR. This method's success hinges on factors such as the quality of the input image, the lighting conditions, and the background complexity.

A. Proposed Method

This study employs OpenCV and Tesseract OCR for license plate detection and recognition, utilizing object detection trained on a specialized dataset of license plate fonts from various regions. Initially, a vehicle image undergoes preprocessing. Should the system spot potential license plate regions or actual plates within the image, it will proceed to segment the characters on the plate. Following character segmentation, OCR technology is applied to recognize and decode each character on the license plate. Subsequent OCR and post-processing methods are then used to enhance and clarify the retrieved license plate data, as illustrated in Figure 10.

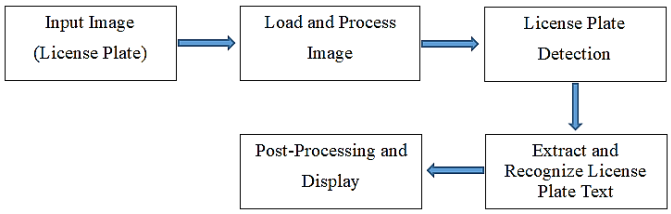


Fig. 10 System Overviews

B. Datasets

For the datasets used in the experiments in this research, we selected 100 license plates, including 50 European plates, especially in the UK, and 50 Lao license plates. We selected those license plates to compare license plate detection and recognition for English characters and Lao characters. All license plate images are “still images” that typically refer to a static image, and the file types are .jpg files of different sizes, as shown in Figure 11.

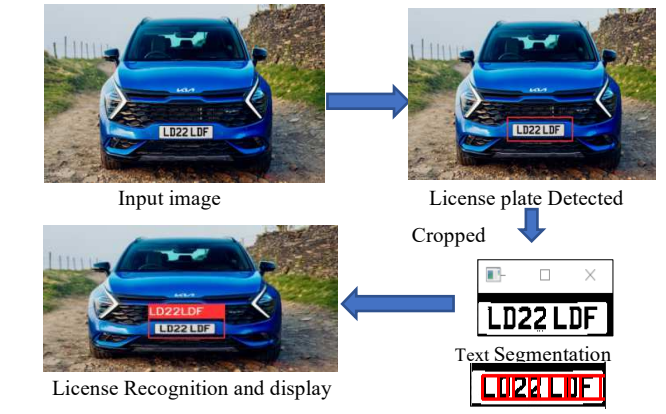
Fig. 11 Vehicle License Plates

The purpose of the license plate used in this experiment was to identify a single line of letters to detect and recognize the letters on the license plate on which each license plate was attached to a different vehicle and environment.

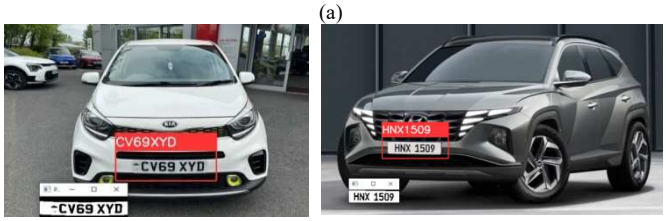
III. RESULTS AND DISCUSSION

The recognition of characters is performed using Tesseract OCR with the required libraries. Then, OpenCV was used with Python to load the image and process it for better plate detection, which may include resizing, converting it to grayscale, and applying various filters. Additional preprocessing, such as blurring and thresholding, is applied if needed to detect regions in the image that may contain license plates. This process can be performed using techniques such as contour detection. The ROI containing the potential license plate was extracted, and Tesseract OCR was used to recognize the text. Finally, post-processing steps, such as filtering out non-alphanumeric characters from the recognized text, are

implemented. The results are displayed in Figure 12, where (a) shows the input image consisting of the vehicle with its license plate detection, cropping, text segmentation and display license plate recognition as ROIs with annotation; (b) shows the detection and recognition of English character license plates from various angles; and (c) shows the detection and recognition of Lao character license plates from various angles.



(a)



(b)

(c)

Fig. 12 Results

From the above results, 100 license plate images were trained by OpenCV and Tesseract OCR. These images were divided into two types: license plates in English and Lao characters, which were tested. After training with the input datasets, the experimental results showed that license plate recognition is faster for English characters than for Lao characters because the Lao language is more complex than English. A comparison of the results of the prediction models produced by OpenCV and Tesseract OCR is shown in Table 2.

TABLE II
COMPARISON OF ENGLISH AND LAO LICENSE PLATE TEMPLATE MATCHING

License Plate Characters	Number of Plates	Plate detected	Cropped License Plate	Processing Time (secs)	Accuracy (%)
English	50	YES	YES	0.12	98.8
Lao	50	YES	YES	0.24	89.42

The table above shows that before recognizing the license plate, some letters may not appear or appear instead of another letter, but the line in the inspection may be the same. The vehicle's position, distance, light, transparency, and sharpness are crucial for license plate detection and recognition. Therefore, in these experiments, the average latency to detect and recognize license plates for English characters was 0.12 s. The average accuracy was 98.8%. For Lao characters, the average latency was 0.24 s, and the average accuracy was 89.42%.

IV. CONCLUSION

Implementing LPR using OpenCV and Tesseract OCR can be a powerful solution for automating tasks related to vehicle identification. Combining OpenCV for image processing and Tesseract OCR for text recognition provides a robust foundation for license plate recognition. The system's accuracy depends on various factors, such as image quality, lighting conditions, and the quality of the OCR engine. OpenCV preprocessing techniques, such as image resizing, color thresholding, and noise reduction, play crucial roles in enhancing the quality of input images and improving the OCR results. ROI detection, such as properly identifying and extracting areas with license plates, is critical for accurate recognition. OpenCV allows for implementing techniques such as contour detection and perspective transformation. The Tesseract OCR can be fine-tuned and configured to improve accuracy, especially for specific fonts and formats commonly used in license plates. Additionally, license plate recognition using OpenCV and Tesseract OCR is a versatile and effective solution with the potential to automate various tasks related to vehicle identification. According to the experimental results, license plate recognition is faster for English characters than for Lao characters because the Lao language is more complex.

ACKNOWLEDGMENT

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (2021R1I1A3058103).

REFERENCES

[1] A. S. D. Sham, P. Pandey, S. Jain, and S. Kalaivani, "Automatic License Plate Recognition Using YOLOV4 and Tesseract OCR," International Journal Of Electrical Engineering and Technology, vol. 12, no. 5, May 2021, doi: 10.34218/ijeet.12.5.2021.006.

[2] I. R. Khan et al., “Automatic License Plate Recognition in Real-World Traffic Videos Captured in Unconstrained Environment by a Mobile Camera,” Electronics, vol. 11, no. 9, p. 1408, Apr. 2022, doi:10.3390/electronics11091408.

[3] W. Puarungroj and N. Boonsirisumpun, “Thai License Plate Recognition Based on Deep Learning,” Procedia Computer Science, vol. 135, pp. 214–221, 2018, doi: 10.1016/j.procs.2018.08.168.

[4] S. Parvin, L. J. Rozario, and Md. E. Islam, “Vehicle Number Plate Detection and Recognition Techniques: A Review,” Advances in Science, Technology and Engineering Systems Journal, vol. 6, no. 2, pp. 423–438, Mar. 2021, doi: 10.25046/aj060249.

[5] O. Bulan, V. Kozitsky, P. Ramesh, and M. Shreve, “Segmentation- and Annotation-Free License Plate Recognition With Deep Localization and Failure Identification,” IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 9, pp. 2351–2363, Sep. 2017, doi: 10.1109/tits.2016.2639020.

[6] A.I. Khan and S. Al-Habsi, “Machine Learning in Computer Vision,” Procedia Computer Science, vol. 167, pp. 1444–1451, 2020, doi:10.1016/j.procs.2020.03.355.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Communications of the ACM, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[8] K. L. Masita, A. N. Hasan, and T. Shongwe, “Deep Learning in Object Detection: a Review,” 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), Aug. 2020, doi:10.1109/icabcd49160.2020.9183866.

[9] A. Akhil, R. R. Praneeth, and Kumar, “Object Detection/Recognition Using Machine Learning Techniques in AWS,” The International Journal of Analytical And Experimental Modal Analysis, vol. 12, no. 3, 2020.

[10] S. N. Srivatsa, G. Sreevathsa, G. Vinay, and P. Elaiyaraja, “Object Detection using Deep Learning with OpenCV and Python,”, International Research Journal of Engineering and Technology (IRJET), vol. 8, no. 1, pp. 227–230, 2021.

[11] Z. Akhtar and R. Ali, “Automatic Number Plate Recognition Using Random Forest Classifier,” SN Computer Science, vol. 1, no. 3, Apr. 2020, doi: 10.1007/s42979-020-00145-8.

[12] Y. Guan et al., “An Object Detection Framework Based on Deep Features and High-Quality Object Locations,” Traitement du Signal, vol. 38, no. 3, pp. 719–730, Jun. 2021, doi: 10.18280/ts.380319.

[13] R. Padilla, S. L. Netto, and E. A. B. da Silva, “A Survey on Performance Metrics for Object-Detection Algorithms,” 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), Jul. 2020, doi: 10.1109/iwssip48289.2020.9145130.

[14] Y. Pan and F. Dong, “Suppression and Enhancement of Overlapping Bounding Boxes Scores in Object Detection,” 2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Dec. 2019, doi:10.1109/isspit47144.2019.9001826.

[15] F. Shao et al., “Deep Learning for Weakly-Supervised Object Detection and Localization: A Survey,” Neurocomputing, vol. 496, pp. 192–207, Jul. 2022, doi: 10.1016/j.neucom.2022.01.095.

[16] X. Wu, D. Sahoo, and S. C. H. Hoi, “Recent advances in deep learning for object detection,” Neurocomputing, vol. 396, pp. 39–64, Jul. 2020, doi: 10.1016/j.neucom.2020.01.085.

[17] W. R. Sania, C. A. Sari, E. H. Rachmawanto, and M. Doheir, “Bounding Box and Thresholding in Optical Character Recognition for Car License Plate Recognition,” sinkron, vol. 8, no. 4, Oct. 2023, doi: 10.33395/sinkron.v8i4.12944.

[18] L. Liu, Y. Wang, and W. Chi, “Image Recognition Technology Based on Machine Learning,” IEEE Access, pp. 1–1, 2024, doi:10.1109/access.2020.3021590.

[19] M. Aamir, “A Progressive Approach to Generic Object Detection: A Two-Stage Framework for Image Recognition,” Computers, Materials & Continua, vol. 75, no. 3, pp. 6351–6373, 2023, doi:10.32604/cmc.2023.038173.

[20] S. Bouraya, and A. Belanngour, “Object Detectors’ Convolutional Neural Networks backbones : a review and a comparative study,” International Journal of Emerging Trends in Engineering Research, vol. 9, no. 11, pp. 1379–1386, Nov. 2021, doi:10.30534/ijeter/2021/039112021.

[21] R. L. Galvez, A. A. Bandala, E. P. Dadios, R. R. P. Vicerra, and J. M. Z. Maningo, “Object Detection Using Convolutional Neural Networks,” TENCON 2018 - 2018 IEEE Region 10 Conference, Oct. 2018, doi: 10.1109/tencon.2018.8650517.

[22] J. Ren, and Yi, W. Shim, “Overview of Object Detection Algorithms Using Convolutional Neural Networks,” Journal of Computer and Communications, vol. 10, no. 1, pp. 115–132, 2022.

[23] A. Kumar and S. Srivastava, “Object Detection System Based on Convolution Neural Networks Using Single Shot Multi-Box Detector,” Procedia Computer Science, vol. 171, pp. 2610–2617, 2020, doi: 10.1016/j.procs.2020.04.283.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” 2014 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 2014, doi: 10.1109/cvpr.2014.81.

[25] G. M. G. Madhuri, B. Shaik, S. S. Tungala, CH. D. D. S. Kumar, and S. V. R. Pilla, “Recognition and Tracing of Object Using CNN,” Journal of Engineering Sciences, vol. 14, no. 03, pp. 589-593, 2023.

[26] R. TH. Hasan and A. . Bibo Sallow, “Face Detection and Recognition Using OpenCV”, jscdm, vol. 2, no. 2, pp. 86–97, Oct. 2021.

[27] M. K. Hossen, "Application of Python-OpenCV to detect contour of shapes and colour of a real image", International Journal of Novel Research in Computer Science and Software Engineering, vol. 9, no. 2, pp. 20-25, 2022.

[28] F. Yu, J. Shuai, B. Yin, and X. Feng, “3D Depth of Field Acquisition Based on OpenCV,” Proceedings of the 2016 2nd Workshop on Advanced Research and Technology in Industry Applications, 2016, doi: 10.2991/wartia-16.2016.195.

[29] A. Shihab, Z. S. Noori, and M. A. Jasim, “Dynamic Object Detection and Evaluation Patterns using OpenCV,” International Refereed Journal of Reviews and Research, vol. 10, no. 1, 2022.

[30] Z. N. Khudhair et al., “Color to Grayscale Image Conversion Based on Singular Value Decomposition,” IEEE Access, vol. 11, pp. 54629–54638, 2023, doi: 10.1109/access.2023.3279734.

[31] M. S. M. Rahim, A. Norouzi, A. Rehman and T. Saba, "3D bones segmentation based on ct images visualization", Biomed. Res., vol. 28, no. 8, pp. 3641-3644, 2017.

[32] J. Chen, B. Guan, H. Wang, X. Zhang, Y. Tang, and W. Hu, “Image Thresholding Segmentation Based on Two Dimensional Histogram Using Gray Level and Local Entropy Information,” IEEE Access, vol. 6, pp. 5269–5275, 2018, doi: 10.1109/access.2017.2757528.

[33] Z. Wang, J. Wang, W. Wang, C. Gao, and S. Chen, “A Novel Thresholding Algorithm for Image Deblurring Beyond Nesterov’s Rule,” IEEE Access, vol. 6, pp. 58119–58131, 2018, doi:10.1109/access.2018.2873628.

[34] J. Shashirangana, H. Padmasiri, D. Meedeniya, and C. Perera, “Automated License Plate Recognition: A Survey on Methods and Techniques,” IEEE Access, vol. 9, pp. 11203–11225, 2021, doi:10.1109/access.2020.3047929.

[35] H. Li, P. Wang, and C. Shen, “Toward End-to-End Car License Plate Detection and Recognition With Deep Neural Networks,” IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 3, pp. 1126–1136, Mar. 2019, doi: 10.1109/tits.2018.2847291.

[36] X. T. Nguyen, K.-T. Nguyen, H.-J. Lee, and H. Kim, “ROI-Based LiDAR Sampling Algorithm in on-Road Environment for Autonomous Driving,” IEEE Access, vol. 7, pp. 90243–90253, 2019, doi: 10.1109/access.2019.2927036.

[37] Y. Wang, X. Zheng, and N. Gao, “A Region of Interest-Based Electrophysiological Source Imaging Technology and its Applications in Analysis of Motor Imagery EEG Signals,” IEEE Access, vol. 11, pp. 140596–140608, 2023, doi:10.1109/access.2023.3339857.

[38] M. Samantaray, A. K. Biswal, D. Singh, D. Samanta, M. Karuppiah, and N. P. Joseph, “Optical Character Recognition (OCR) based Vehicle’s License Plate Recognition System Using Python and OpenCV,” 2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Dec. 2021, doi:10.1109/iceca52323.2021.9676015.

[39] C. Adjetey and K. S. Adu-Manu, “Content-based Image Retrieval using Tesseract OCR Engine and Levenshtein Algorithm,” International Journal of Advanced Computer Science and Applications, vol. 12, no. 7, 2021, doi:10.14569/ijacsa.2021.0120776.

[40] S. Bansal, M. Gupta, and A. K. Tyagi, “A Necessary Review on Optical Character Recognition (OCR) System for Vehicular Applications,” 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Jul. 2020, doi:10.1109/icirca48905.2020.9183330.

[41] T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait, “High-Performance OCR for Printed English and Fraktur Using LSTM Networks,” 2013 12th International Conference on Document Analysis and Recognition, Aug. 2013, doi: 10.1109/icdar.2013.140.

[42] X. F. Wang, Z.-H. He, K. Wang, Y.-F. Wang, L. Zou, and Z.-Z. Wu, “A survey of text detection and recognition algorithms based on deep learning technology,” Neurocomputing, vol. 556, p. 126702, Nov. 2023, doi: 10.1016/j.neucom.2023.126702.