DOI: 10.31276/VJSTE.2023.0099

An integration of segmentation technique on edge devices for license plate recognition

Duy Dieu Nguyen[1*], Tan Sang Vo[2], Manh Hung Le[3], Minh Son Nguyen[3]

[1]University of Management and Technology, Ho Chi Minh City, Road 60CL, Cat Lai Urban Area, Thu Duc City, Ho Chi Minh City, Vietnam
[2]University of Transport Ho Chi Minh City, 2 Vo Oanh Street, Ward 25, Binh Thanh District, Ho Chi Minh City, Vietnam
[3]University of Information Technology, Vietnam National University - Ho Chi Minh City, Quarter 6, Linh Trung Ward, Thu Duc City, Ho Chi Minh City, Vietnam

Received 26 October 2023; revised 11 April 2024; accepted 11 July 2024

Abstract:

Typically, vehicle license plate recognition involving large quantities of images is carried out centrally in data centre. This results in high infrastructure and operational costs and presents difficulties for widespread deployment. To address these limitations, we propose a solution that deploys license plate recognition algorithms on edge computing devices, reducing the load on both infrastructure and centralised systems. For the purpose of training the license plate recognition model, we gathered more than 5,000 images of vehicles from various street and parking lot environments. We employed YOLOv8 for segmenting license plates and recognising the characters. Following segmentation, point sets were obtained, and based on these point sets, the license plate was reoriented to a frontal view. This allowed us to achieve a recognition accuracy of 99.21% in identifying license plate characters. Testing results on a Jetson Nano device, using 640x640 resolution data under different lighting and weather conditions, revealed an average processing speed of approximately 2.2 fps. In particular, we successfully segmented and classified license plates at distances ranging from 0.5 to 3 m, with an accuracy of up to 99.53%. This method is highly efficient, with low computational costs, and allows for smooth operation on embedded devices without compromising accuracy when compared to commercial systems.

Keywords: **AIoT, edge computing, license plate on embedded devices, license plate segmentation, license plate recognition.**

Classification numbers: **1.2, 1.3**

1. Introduction

Various models and machine learning algorithms exist for the segmentation and recognition of license plates. However, these models face challenges such as dirt, distortion, damage, and lighting conditions, all of which can degrade the quality of license plate images. Furthermore, the system's implementation is also a crucial factor to consider. When deploying an Automatic License Plate Recognition (ALPR) system [1] that utilises a centralised processing system, the images captured by cameras are sent to a central server for analysis and processing. Transmitting large volumes of image data to a central processing server significantly affects the system's real-time responsiveness. This process demands considerable processing capacity and power from the central system to meet performance requirements. Moreover, transmitting unprocessed image data or raw data from numerous deployment locations to the central server necessitates high network bandwidth, making the deployment, maintenance, and development of such systems prohibitively expensive.

To overcome the aforementioned limitations, we propose the use of edge computing. The utilisation of embedded devices and edge computing in license plate recognition reduces the load on the central system, thereby improving the user experience. These systems can be flexibly integrated into parking lots, vehicle tracking systems, or smart city solutions. By processing information locally on the edge device, independent of the central system, this approach

[*]*Corresponding author: Email: dieu.nguyenduy@umt.edu.vn*

enhances real-time processing capabilities and minimises potential errors. Furthermore, data exchange between the central system and the device is minimised, as only processed information is transmitted to the central system for storage. This processed data is also compressed, resulting in lighter encryption requirements and enhanced security, all while requiring minimal network traffic. Additionally, the compact nature of embedded devices makes them easy to deploy when needed. Overall, this approach offers numerous benefits and improves the experience for both managers and users (Fig. 1).

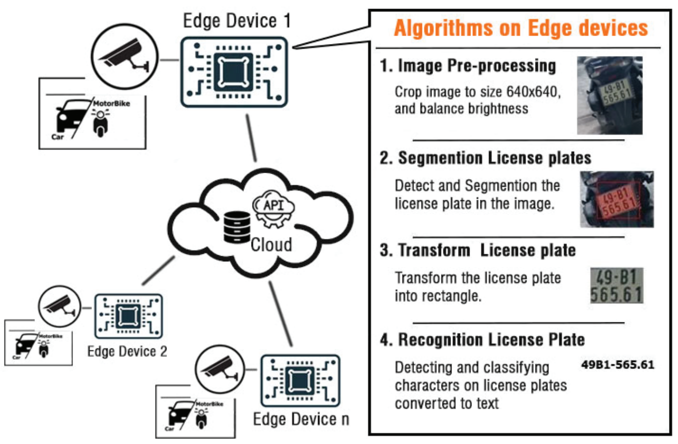

Edge Device 1
MotorBike
Car
Algorithms on Edge devices
1. Image Pre-processing
Crop image to size 640x640, and balance brightness
2. Segmention License plates
Detect and Segmention the license plate in the image.
3. Transform License plate
Transform the license plate into rectangle.
49-B1 565.61
4. Recognition License Plate
Detecting and classifying characters on license plates converted to text
49B1-565.61
API
Cloud
Edge Device 2
Edge Device n

Fig. 1. A solution for license plate recognition based on edge devices.

While the use of embedded devices and edge computing for ALPR offers many advantages, it still presents challenges. The most significant challenge is the processing performance of the device used. Most embedded devices have limited computational power, but when deployed, the volume of ALPR processing required is substantial. This means that the ALPR model must be optimised to meet the system's real-time processing and energy efficiency requirements. Numerous studies have explored the deployment of license plate recognition algorithms on embedded devices, including GAN-based algorithms, the pix2pix model, DNAS, Neural Blocks in Search Space, and search strategies introduced in [2]. However, most of these studies encounter difficulties when attempting to detect license plates that are skewed relative to the camera's orientation.

In this article, we propose an optimised ALPR model for license plate recognition, which uses a segmentation algorithm to reorient skewed license plates to a frontal view. This approach effectively addresses the common issues faced by other algorithms, such as tilted or misaligned license plates. Our model is designed for deployment on embedded and edge computing devices. Various segmentation algorithms can be employed for this task; in this study, we chose to use YOLOv8 [3] to implement our proposed model. This was done using a dataset of license plate images that we collected, classified, and annotated for training and testing purposes.

In the following sections, we present the research and the specific results obtained. Section 2 provides an overview of the related work in license plate segmentation and recognition, the applications, and the specific challenges posed by embedded devices and edge computing. Section 3 presents the proposed methods for segmenting and recognising license plates. Section 4 outlines the experimental process and the system's results. In Section 5, we discuss the practical applications and possible future extensions. Finally, Section 6 concludes the paper with a summary of our findings and suggests future research directions.

2. Related works

Segmentation and recognition of license plates on embedded devices and edge computing have become key areas of research in vehicle identification. Given the limited processing resources of embedded devices, optimising computational performance is essential for efficient image processing. The use of edge computing provides significant advantages, optimising performance, conserving energy, facilitating system integration, and enhancing security. Furthermore, edge computing reduces data processing latency, making it suitable for real-time applications. In this section, we review related studies to identify the challenges and practical solutions relevant to the proposed method.

2.1. Method of detecting license plates

2.1.1. Traditional machine learning method

Several traditional methods for detecting license plates have been proposed, using conventional image processing techniques. K. Deb, et al. (2008) [4] proposed detecting license plates through colour transformation and geometric modifications, while A.M.A. Ghaili, et al. (2013) [5], also utilised similar techniques. M.T. Qadri, et al. (2009) [6] used boundary edges in images to segment the license plate area. These methods aim to extract license plate regions based on differences in colour, geometric properties, and edge information. However, in practice, varying environmental lighting conditions and differences in license plate designs across countries can lead to low sensitivity, limiting the effectiveness of these approaches.

2.1.2. Neural network-based approach

To overcome the limitations of traditional methods, researchers have shifted towards machine learning and deep learning techniques. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have shown promising results in license plate recognition and segmentation tasks [7, 8]. CNNs offer high discriminative power and are used to extract license plate features, while BLSTM (Bi-Directional Long Short-Term Memory) can capture contextual information from past data [9]. Although these methods are effective, they have high computational costs, requiring devices with substantial processing power to function optimally.

2.2. License plate segmentation technique

2.2.1. Traditional image processing methods

Traditional image processing techniques have also been extensively studied for license plate segmentation. The edge-based approach utilising Canny edge detection and morphological operations to identify license plate regions is one such method. K.R. Soumya, et al. (2014) [10] focused on adaptive thresholding and contour analysis for accurate segmentation.

2.2.2. Deep learning-based approach

More recently, researchers have explored the use of deep learning techniques for license plate segmentation. X. Tian, et al. (2022) [11] introduced a CNN based on the GRU (Gated Recurrent Unit) sequence model, achieving high accuracy in various datasets. Similarly, R.J. Tom, et al. (2022) [12] introduced a modified U-Net model that improves segmentation performance, particularly in challenging lighting conditions.

2.3. License plate recognition technique

2.3.1. Traditional machine learning method

Traditional machine learning methods have also been applied to license plate recognition. R.F. Prates, et al. (2013) [13] used graph-oriented gradients and support vector machines (SVMs) for character recognition, achieving competitive results on a benchmark dataset.

2.3.2. Deep learning-based approach

Deep learning has demonstrated considerable success in license plate recognition tasks. S.G. Kim, et al. (2017) [14] proposed a vehicle zone detection-based approach that incorporates scale information and search range limitations for reliable license plate zone detection. In addition, J. Shashirangana, et al. (2021) [15] explored neural networks (NAS) for license plate detection and recognition, specifically FBNet and PC-DARTS, to optimise the model. However, partially obscured license plates or extremely bright lighting conditions still pose challenges for accurate identification.

2.4. Devices embedded in license plate segmentation and recognition

2.4.1. NVIDIA Jetson family of devices

NVIDIA Jetson is a GPU-enabled embedded device platform designed for AI tasks [16]. Due to its multi-dimensional processing power, many researchers have developed artificial intelligence applications, with license plate recognition being one of them. N. Awalgaonkar, et al. (2021) [17] used the Mobilenet V1 architecture based on a Single Shot Detector (SSD) for license plate localisation. The proposed system, implemented on the Jetson Nano, enables video processing for ALPR with an accuracy of over 95%.

2.4.2. Raspberry Pi family of devices

Raspberry Pi is a simple, low-cost microprocessor device [18] widely used in various application areas, including embedded computing, monitoring systems, and IoT. Researchers have demonstrated that it is possible to successfully implement license plate segmentation and recognition on Raspberry Pi boards. A.O. Agbeyangi, et al. (2020) [19] developed a lightweight model optimised for Raspberry Pi, enabling real-time processing on these low-cost devices.

Although the aforementioned studies have made significant progress in license plate segmentation and recognition, several challenges remain. One major challenge is the variation in license plate designs, fonts, and backgrounds across different countries. Another challenge is ensuring the system's ability to adapt to various lighting and weather conditions, as well as different camera angles. In summary, license plate segmentation and recognition on embedded devices and edge computing have been explored using a range of techniques, from traditional image processing to machine learning and deep learning methods. Advances in these fields have paved the way for scalable and real-time systems, although further research is needed to improve robustness and adaptability in real-world scenarios.

3. System model and methods

3.1. Model overview

The license plate recognition system consists of five stages: license plate detection, license plate segmentation, alignment, character segmentation, and character recognition. Typically, complex image processing algorithms that result in significant delays are used for license plate detection and character recognition. However, we utilise YOLOv8 Nano for license plate detection and segmentation, followed by character recognition using edge computing to minimise the load on the centralised system.

License plate recognition begins when an edge computing device, containing an algorithmic model (Edge AI), receives an image from a camera. The algorithm recognises the position of the license plate within the image, preprocesses the license plate area, and then recognises the characters on it. To improve accuracy, we trained the model using the formats of Two-line and One-line license plates from Vietnam. This method enhances accuracy, as the Edge AI is able to distinguish between lines without confusion, delivering quick results.

3.2. Construct and label datasets

3.2.1. Segmentation license plates dataset

This dataset comprises data on car and motorbike license plates from Vietnam. The data were collected using two methods: extracting from online sources and collecting directly in natural environments, encompassing a range of environmental conditions, weather, time, and shooting angles. We classified the dataset into two categories: One-line license plates (LpD) and Two-line license plates (LpV) (Fig. 2). Each class includes three types of license plates: yellow license plates (for transport vehicles); blue license plates (for government agencies); white license plates (for private vehicles).

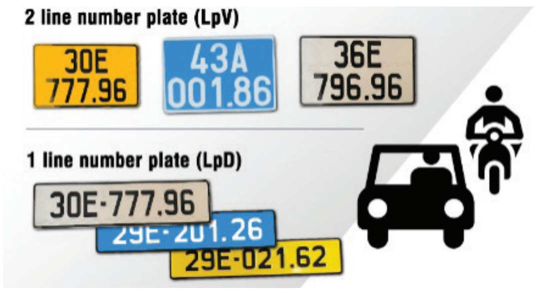


Fig. 2. Types of One-line and Two-line license plates in Vietnam.

We shared this dataset publicly [20]. In addition, we also collected data on red (military) license plates. However, due to the sensitive nature of this type of license plate, we are only using it internally and will not share it publicly.

After data collection, we labelled the images using the LabelMe application, marking the license plate area with surrounding boundaries (Polygon). The dataset contains approximately 5,000 images, with 3,500 samples of One-line license plates and 1,625 samples of Two-line license plates. To ensure data diversity, the dataset was divided into 70% for training, 25% for evaluation, and 5% for testing.

3.2.2. Recognising license plate characters dataset

We collected a character dataset from license plates, as shown in Fig. 3, following the steps of segmentation (step A), alignment (step B), and character splitting (step C).

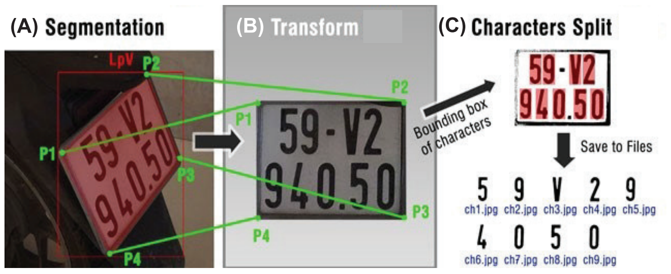


Fig. 3. Steps to build the license plate character dataset. (A) Segmentation, **(B)** Transform, **(C)** Character splitting.

To reframe the shape of the license plate, we used points P1, P2, P3, and P4 to segment (step A) and separate the license plate image. Then, by aligning the corners of the license plate using four corner coordinates (step B), we transformed the plate into a rectangular shape with specific dimensions. This ensures that the license plate is displayed directly.

Next, to locate the characters, we used contour detection to identify object boundaries. Objects with a height of approximately 30-40% of the license plate and a width of 20-40% were selected. We then saved the image data of these bounding boxes.

After processing all the license plates, we applied the k-means algorithm to group characters with similar morphology, filtered out errors, removed duplicate or noisy images, and labelled the groups. This resulted in 32,933 image files across 37 character classes, which include letters from A to Z, numbers from 0 to 9, and space characters.

3.3. YOLOv8 model for license plate segmentation and character detection

In this study, we used YOLOv8 to perform license plate segmentation and detect the positions of characters. YOLOv8 comes in various versions - nano, small, medium, large, and extra-large - each designed for different purposes, ranging from compact models to larger systems. As the number of parameters increases, model accuracy improves, though speed decreases. Detailed information on the parameters is presented in Table 1.

Table 1. Params and flops of YOLOv8.

YOLOv8 model	Picture size (W&H)	Detection		Segmentation	
		Params (M)	*FLOPs (B)*	*Params (M)*	*FLOPs (B)*
Nano	640	3.0	8.7	3.2	12.6
Small	640	11.2	28.6	118.	42.6
Medium	640	25.9	78.9	27.3	110.2
Large	640	43.7	165.2	46	220.5
Xtra-large	640	68.2	257.8	71.8	344.1

We conducted training on hardware equipped with a Tesla NVIDIA T4 GPU (16GB RAM and 2560 NVIDIA CUDA® Cores) to train both the segmentation and character recognition models. For the segmentation model, we trained for 300 epochs with a batch size of 64, using image sizes of 640x640 pixels. Each epoch took 1 minute and 45 seconds, and the best result (99.53% accuracy) was achieved at epoch 249. For the character recognition model, we trained for 300 epochs with a batch size of 128, using an image size of 240 pixels. Each epoch averaged 45 seconds, with the best result (99.21% accuracy) at epoch 281.

3.4. The process of recognising license plates

After data collection, training, and obtaining both segmentation and detection models, we deployed them onto an embedded device. The process follows six steps to output the text data of the license plate found in the image (Fig. 4).

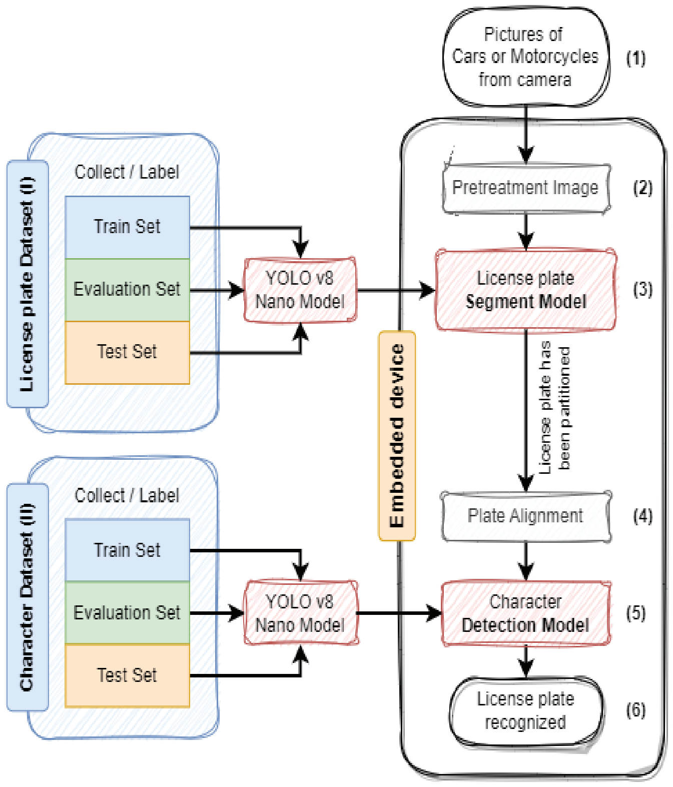


Fig. 4. Diagram of license plate recognition.

3.4.1. Input data acquisition

The clearest image is selected from 10 consecutive frames using a blur confirmation algorithm. Once the clearest image is identified, it proceeds to the next step for processing.

3.4.2. Image pre-processing

The brightness of the image is adjusted, and its size is resized without altering the original aspect ratio. The smallest dimension is set to 640 pixels. From the image centre, the image is cropped to 640x640 pixels for input into the license plate segmentation model.

3.4.3. Segmentation of the license plate location

YOLO's segmentation model is used to detect the license plate region (Fig. 5). The output includes polygon shapes representing One-line (LpD) or Two-line (LpV) license plates, created from sets of points (x, y).

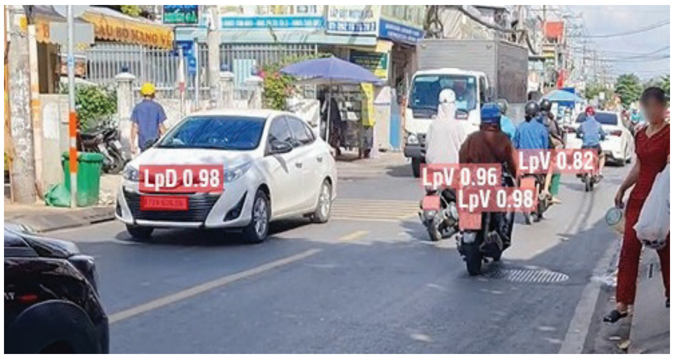


Fig. 5. Real-time segmentation of license plates on a street.

3.4.4. License plate alignment

From the polygonal points obtained in step 3, we identify the four corner points (P1, P2, P3, P4) of the license plate. We find these points by forming a convex polygon with the longest sides (Fig. 6).

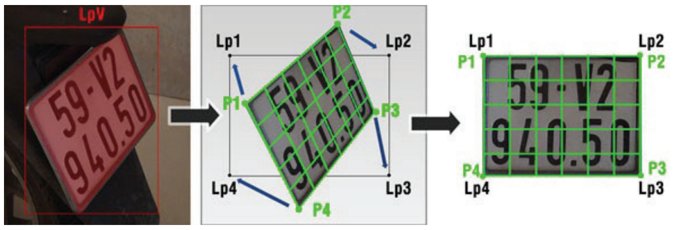


Fig. 6. Transformation (alignment) of the license plate.

When the coordinates of the corner points of the license plate (P1, P2, P3, P4) are obtained, we perform image transformations such as translation, rotation, deformation, affine, and coordinate transformations. This is done by multiplying the matrix with the homography matrix [21]. The four points - P1, P2, P3, and P4 - are transformed into a rectangular image with the coordinates: Lp1(0, 0), Lp2 (width, 0), Lp3 (width, height), and Lp4 (0, height). The width and height of this rectangle depend on the type of license plate (LpV or LpD) detected in the previous step. The mathematical formula for this function is described as follows:

We define the matrix P, which extends from the input and destination points, as shown in Eq. (1):

$$P = \begin{vmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1.x_1' & -y_1.x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1.y_1' & -y_1.y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2.x_2' & -y_2.x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2.y_2' & -y_2.y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3.x_3' & -y_3.y_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3.y_3' & -y_3.y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4.x_4' & -y_4.y_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4.y_4' & -y_4.y_4' \end{vmatrix} \quad (1)$$

Here, (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) are the coordinates of the points in the original image (P1, P2, P3, and P4, respectively), and (x_1', y_1'), (x_2', y_2'), (x_3', y_3'), and (x_4', y_4') are the coordinates of the points in the target rectangular image (Lp1, Lp2, Lp3, and Lp4, respectively).

Next, using the expanded matrix A, we solve the following equation to calculate the transformation matrix, as shown in Eq. (2):

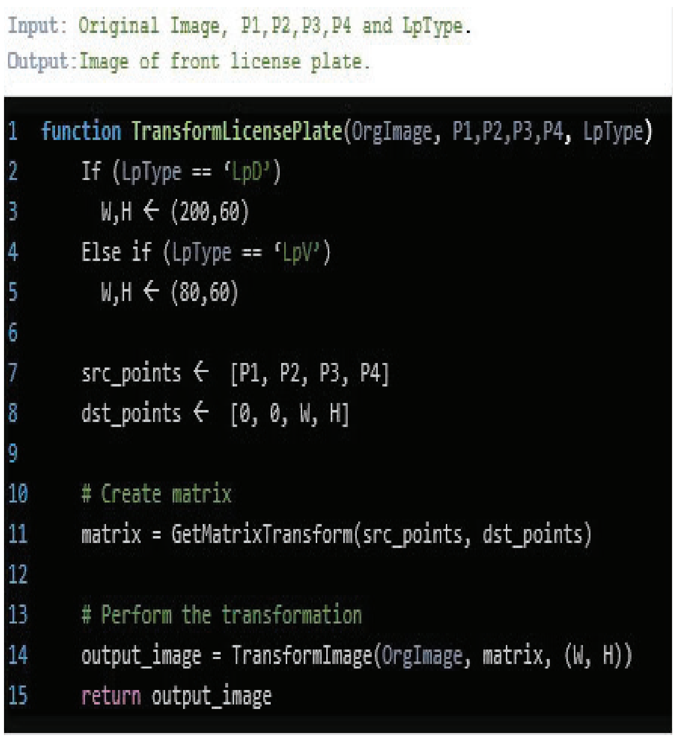
$$P.M=0 \quad (2)$$

The transformation matrix M is a 3x3 matrix and is shown in Eq. (3):

$$M = \begin{vmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{vmatrix} \quad (3)$$

The elements of matrix M are determined through the equations in the expanded matrix A. This is how the transformation matrix is calculated to transform an object in an image from quadrilateral $P1$, $P2$, $P3$, $P4$ to rectangular Lp1, Lp2, Lp3, Lp4.

The result of the change is an aligned license plate image. See Algorithm 1 for more details on transforming the license plate to a frontal view.

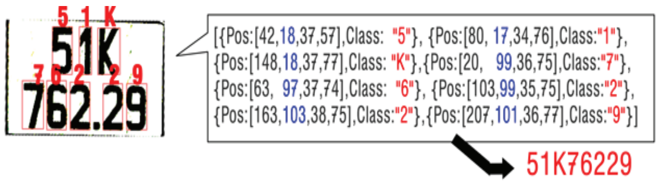


```
Input: Original Image, P1,P2,P3,P4 and LpType.
Output:Image of front license plate.

function TransformLicensePlate(OrgImage, P1,P2,P3,P4, LpType)
    If (LpType == 'LpD')
      W,H ← (200,60)
    Else if (LpType == 'LpV')
      W,H ← (80,60)

    src_points ←  [P1, P2, P3, P4]
    dst_points ←  [0, 0, W, H]

    # Create matrix
    matrix = GetMatrixTransform(src_points, dst_points)

    # Perform the transformation
    output_image = TransformImage(OrgImage, matrix, (W, H))
    return output_image
```

Algorithm 1. Transformation of the license plate image.

3.4.5. Recognition of characters on the license plate

After the license plate calibration process, a direct image of the license plate is obtained, reducing distortion and improving character detection accuracy. This leads to more precise character recognition and classification. The image processing algorithm returns a JSON array {x, y, width, height, class}, as shown in Fig. 7.



```
[{Pos:[42,18,37,57],Class: "5"}, {Pos:[80, 17,34,76],Class:"1"},
{Pos:[148,18,37,77],Class: "K"},{Pos:[20,  99,36,75],Class:"7"},
{Pos:[63,  97,37,74],Class:  "6"}, {Pos:[103,99,35,75],Class:"2"},
{Pos:[163,103,38,75],Class:"2"},{Pos:[207,101,36,77],Class:"9"}]
```

51K76229

Fig. 7. An example character recognition on a license plate.

3.4.6. Determining license plate content

Based on the x-coordinates, we determine the order of appearance of the characters, and based on the y-coordinates, we identify the line arrangement. As illustrated in Fig. 7, the red characters correspond to the y-coordinates. For instance, coordinates 17 and 18 are from one line, and coordinates 97, 99, 101, and 103 are from another line. The final result is the text extracted from the license plate (Fig. 8).

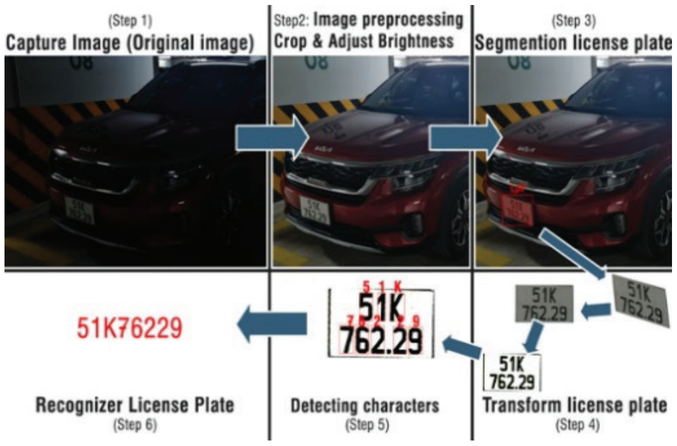


Fig. 8. The six-step process of recognising a license plate.

3.5. Model evaluation

To evaluate the model on embedded systems, we used a dataset of 1,139 images, comprising 903 Two-line license plates and 410 One-line license plates, taken under various environmental conditions, including day-time, night-time, heavy rain, and sunny weather, as well as at different angles on the street. The test results are summarised as follows: Correct detection refers to the number of license plates that were correctly identified. False detection occurs when one or more characters are incorrectly identified, leading to an overall incorrect result. Unrecognised plates are those that could not be recognised by the model. These results are summarised in Table 2.

Table 2. Evaluation results for license plate recognition.

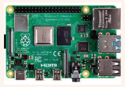
License plate type	Correct detections	False detections	Unrecognised plates
Two-line plates	892	9	2
One-line plates	406	1	3

In cases where license plates were incorrectly detected, the majority of errors were due to damage or excessive obstruction, while unrecognisable plates were often caused by extreme warping.

4. Experiments and results

After training on a dataset of more than 5,000 images collected from natural street conditions (as mentioned in Section III), we conducted tests on both general-purpose and embedded devices to evaluate performance under real-time conditions. The tests were carried out on one desktop computer and two embedded devices: Jetson Nano B01 and Raspberry Pi 4 (purchased from Amazon in April 2023). The configuration details of the devices are presented in Table 3.

Table 3. Configuring hardware devices in the experiment.

Device	Information	Cost
RASPBERRY PI4 MODEL B 	CPU: Quad core -A72 64-bit SoC @ 1.5GHz GPU: None RAM: 4GB	59.95 USD
JETSON NANO	CPU: Quad-core 64-bit ARM @1.43Ghz GPU: 128 Core-Maxwell RAM: 4GB	209.00 USD
DESKTOP NUC	CPU: Intel® Core™ i7-8850H @2.60GMhz GPU: Intel® UHD 630 RAM: 16GB	597.82 USD
CAMERA HIK Vision	USB Camera 720p Optical image stabilisation	42.00 USD

4.1. Results

4.1.1. License plate segment model

To evaluate the accuracy of license plate classification, we used the mean Average Precision (mAP) method. The mAP value depends on both precision and recall data. AP (Average Precision) is calculated by summing the area under the Precision-Recall curve for each feature class, and the mAP is the average of these APs. The formula for calculating mAP is shown in Eq. (4) [22]:

$$\mathrm{mAP} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{AP_i} \tag{4}$$

The results were impressive, with the dataset yielding an average value for the license plate detection and segmentation model of 99% for the two types of license plates: square (LpV) and long (LpD), as shown in Fig. 9. The mAP@50 (average of the first 50 AP values) achieved a maximum result of 99.35%.

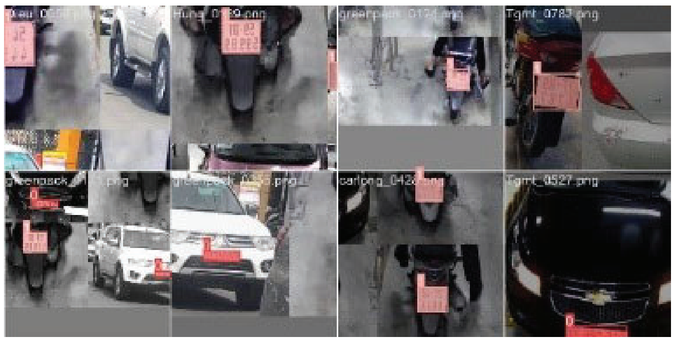


Fig. 9. Sample images obtained from the model.

4.1.2. License plate recognition model

We used various datasets to experiment with the model, covering distances ranging from 50 to 300 cm, under different weather and daylight conditions. These datasets were categorised such that each contained 100 license plates within the specified distance range, with each image potentially containing one or more license plates. License plates that did not meet the distance criteria were excluded. The results were evaluated using the standard F1-score and are summarised in Table 4.

Table 4. Experiment results under environmental conditions.

Distance (cm)	During the day (%)	In the rain (%)	Night on the street (%)
50-100	100	100	100
150-200	100	98	95
250-300	97	93	86

In real-world testing under different lighting and weather conditions, we observed that the accuracy decreased at greater distances. When the camera failed to capture clear license plate pixels, recognition accuracy suffered, leading to false identifications or unrecognisable results. At night, reflections from street lighting further reduced accuracy when using a standard camera (Fig. 10).

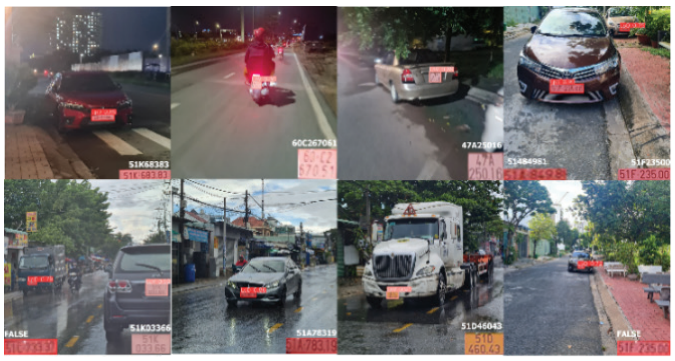


Fig. 10. Sample images from the experimental results.

4.2. Experimenting with steps performed on embedded devices

In this test, we selected and collected additional images under varying conditions (distance, daylight, night, and weather, including sunny and rainy conditions).

4.2.1. Measurement of time

We used the prepared dataset to test the processing time on Raspberry Pi 4, Jetson Nano B01, and Desktop NUC devices. Each device was tested 10 times, and the processing speed for each image was recorded. The average processing time for each step (segmentation, identification, and others) is shown in Table 5.

Table 5. Processing time of algorithms.

	Raspberry Pi 4 (ms)	Jetson Nano B01 (ms)	Desktop NUC (ms)
Segmentation (Step 3)	2,346	221	161
Recognition (Step 5)	1,750	162	99
Other (Steps 1, 2, 4, 6)	110	82	43
All	4,206	465	303

The results (Table 5) indicate that from input image processing (step 1) to character recognition (step 6), the process was smooth. The real-time test on Raspberry Pi 4 revealed that processing one frame took 4,206 ms (equivalent to 0.24 fps). This is relatively slow because the Pi 4 does not have GPU support, making real-time processing challenging. In contrast, the desktop computer processed a frame in just 303 ms (equivalent to 3.3 fps), and Jetson Nano processed a frame in 465 ms (equivalent to 2.2 fps). The processing speeds of both the desktop and Jetson Nano were fast enough to meet the real-time requirements for edge device applications.

To assess the overall response time of the devices, we conducted tests on each task 20 times and calculated the average time. These tasks included model boot time and camera-device communication time. Detailed results are provided in Table 6.

Table 6. Response time analysis.

	Raspberry Pi 4 (ms)	Jetson Nano B01 (ms)	Desktop NUC (ms)
First time processing the algorithm	12,637	8,392	6,128
Camera response for the first time	1655	1461	814
Camera response	130	82	73

4.2.2. Distance testing

To develop a solution for license plate segmentation and recognition in parking management systems, we tested distances that reflected real-world parking conditions, ranging from 0.5 to 3 m. Beyond 3 m, recognition results deteriorated due to insufficient pixel information for identification. The test results for the embedded devices are presented in Fig. 11.

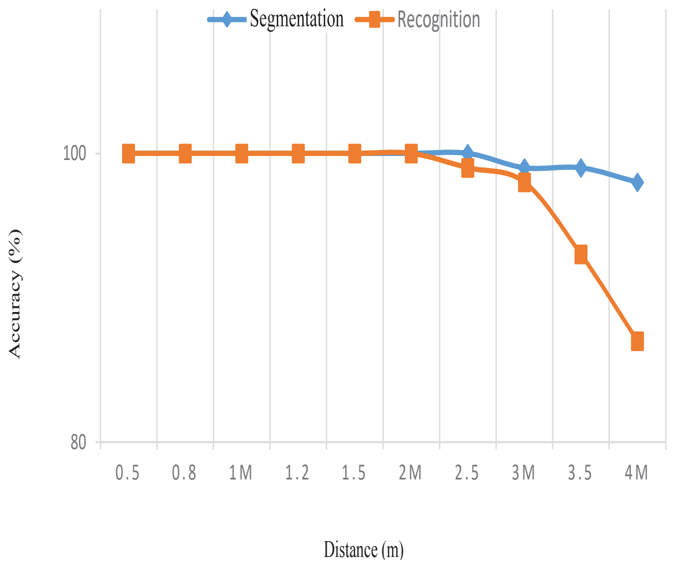


Fig. 11. Recognition distance between edge device and license plate.

The results from the test intervals, using the models and equipment tested, met the required standards. Detection and recognition of license plates from 50 to 250 cm achieved optimal results. Although embedded devices have limited processing power, the results were positive. With this working distance, the system is well-suited for parking lot applications and limits errors or failed detections due to wide camera angles (greater than 75°).

5. Discussion

Utilising segmentation algorithms on edge computing devices for license plate recognition offers several advantages, making it highly suitable for practical applications. Firstly, by shifting the identification process from centralised data centres to edge devices, infrastructure and operational costs are significantly reduced. This not only decreases the burden on central systems but also facilitates widespread deployment, potentially enhancing the accessibility and scalability of license plate recognition technology.

Secondly, the high accuracy achieved in recognising number plate characters - reaching 99.21% - is noteworthy. The system's ability to recognise multiple number plates simultaneously, from various angles, and under different lighting conditions is also a significant strength. Furthermore, the ability to accurately segment and classify license plates at distances ranging from 0.5 to 3 m emphasises the practical applicability of the proposed method in real-world scenarios. Finally, the efficiency of this method, combined with its low computational cost, is particularly notable. With an average processing speed of 2.2 fps on the Jetson Nano and the ability to maintain high accuracy, this method proves suitable for deployment on embedded devices with limited computational resources. This not only enhances operational efficiency but also opens opportunities for deployment in resource-constrained environments, where traditional centralised systems may not be practical.

However, despite these strengths, there are limitations that require improvement. It is necessary to conduct a more comprehensive evaluation of the model's performance across a broader range of scenarios and datasets, such as those involving infrared cameras or dual cameras. Recognition distance also needs further attention, as it influences the clarity of the license plate in the captured image. Additionally, in situations with very weak lighting, the system sometimes fails to process images due to insufficient data. Incorporating an infrared camera could help to address this issue. Although the reported results are promising, further testing under varied conditions would provide a more robust assessment of the model's generalisability.

In summary, the proposed solution presents a promising approach to license plate recognition, leveraging edge computing devices to achieve high efficiency. This solution can be applied to the deployment of vehicle tracking and enforcement systems based on license plate numbers across large geographic areas. While there are several areas that require further improvement and evaluation, this approach represents an important step towards enabling the widespread deployment of license plate recognition technology in diverse real-world environments.

6. Conclusions

In this study, we focused on the development and application of vehicle license plate recognition technology through image processing, primarily using embedded devices and edge computing systems. Our method demonstrated high efficiency and impressive accuracy in segmenting and classifying license plates. The study highlights that this method operates smoothly on embedded devices with low computational costs, which is essential when applying license plate recognition technology to resource-constrained applications, such as distributed smart parking systems or automatic toll collection systems on highways.

The test results also show that our method performs effectively under various weather and lighting conditions, with an average processing speed of approximately 2.2 fps when using 640x640 pixel resolution data on the Jetson Nano. The method achieved an accuracy of 99.53% in segmenting and classifying license plates at distances between 0.5 and 3 m, while the accuracy for recognising the content of the license plates reached 99.21%. Additionally, we contributed a dataset of 5,000 segmented license plate images from vehicles in Vietnam, along with 32,933 character images from classified license plates. This dataset will help advance research in license plate detection and recognition, particularly on embedded and real-time devices. This study can serve as a valuable reference for future efforts in the development and improvement of real-time license plate recognition technology on embedded devices.

CRediT author statement

Duy Dieu Nguyen: Conceptualisation, Supervision, Methodology, Data collection and processing, Writing - Original draft preparation; Tan Sang Vo: Data collection and processing, Writing - Original draft preparation; Manh Hung Le: Conceptualisation, Methodology, Data collection and processing, Writing - Original draft preparation; Minh Son Nguyen: Conceptualisation, Methodology, Professional advisor, Reviewing and Editing.

COMPETING INTERESTS

The authors declare that there is no conflict of interest regarding the publication of this article.

REFERENCES

[1] T. Vaiyapuri, S.N. Mohanty, M. Sivaram, et al. (2021), "Automatic vehicle license plate recognition using optimal deep learning model", *Computers, Materials & Continua*, **67(2)**, pp.1881-1897, DOI: 10.32604/cmc.2021.014924.

[2] H. Padmasiri, J. Shashirangana, D. Meedeniya, et al. (2022), "Automated license plate recognition for resource-constrained environment", *Sensors*, **22(4)**, DOI: 10.3390/s22041434.

[3] Ultralytics (2023), *Ultralytics*, https://docs.ultralytics.com/, accessed 15 August 2023.

[4] K. Deb, K.H. Jo (2008), "HSI color based vehicle license plate detection", *2008 International Conference on Control, Automation and Systems*, pp.687-691, DOI: 10.1109/ICCAS.2008.4694589.

[5] A.M.A. Ghaili, S. Mashohor, A.R. Ramli, et al. (2013), "Vertical-edge-based car-license-plate detection method", *IEEE Transactions on Vehicular Technology*, **62(1)**, pp.26-38, DOI: 10.1109/TVT.2012.2222454.

[6] M.T. Qadri, M. Asif (2009), "Automatic number plate recognition system for vehicle identification using optical character recognition", *2009 International Conference on Education Technology and Computer*, pp.335-338, DOI: 10.1109/ICETC.2009.54.

[7] H. Li, P. Wang, C. Shen (2018), "Toward end-to-end car license plate detection and recognition with deep neural networks", *IEEE Transactions on Intelligent Transportation Systems*, **20(3)**, pp.1126-1136, DOI: 10.1109/TITS.2018.2847291.

[8] N. Saif, N. Ahmmed, S. Pasha, et al. (2019), "Automatic license plate recognition system for Bangla license plates using convolutional neural network", *TENCON 2019 - 2019 IEEE Region 10 Conference*, pp.925-930, DOI: 10.1109/TENCON.2019.8929280.

[9] P. Shivakumara, D. Tang, M. Asadzadehkaljahi, et al. (2017), "CNN-RNN based method for license plate recognition", *Selected Papers from The 4th Asian Conference on Pattern Recognition (ACPR 2017)*, **3(3)**, pp.169-175, DOI: 10.1049/trit.2018.1015.

[10] K.R. Soumya, A. Babu, L. Therattil (2014), "License plate detection and character recognition using contour analysis", *International Journal of Advanced Trends in Computer Science and Engineering*, **3(1)**, pp.15-18.

[11] X. Tian, L. Wang, R. Zhang (2022), "License plate recognition based on CNN", *2022 14th International Conference on Computer Research and Development (ICCRD)*, pp.244-249, DOI: 10.1109/ICCRD54409.2022.9730272.

[12] R.J. Tom, A. Kumar, S.B. Shaik, et al. (2022), "Car license plate detection and recognition using modified U-net deep learning model", *2022 8th International Conference on Smart Structures and Systems (ICSSS)*, pp.1-6, DOI: 10.1109/ICSSS54381.2022.9782176.

[13] R.F. Prates, G.C. Chávez, W.R. Schwartz, et al. (2014), "Brazilian license plate detection using histogram of oriented gradients and sliding windows", *International Journal of Computer Science & Information Technology (IJCSIT)*, **5(6)**, pp.39-52, DOI: 10.5121/ijcsit.2013.5603.

[14] S. Kim, H. Jeon, H. Koo (2017), "Deep-learning-based license plate detection method using vehicle region extraction", *Image and Vision Processing and Display Technology*, **53(15)**, pp.1034-1036, DOI: 10.1049/el.2017.1373.

[15] J. Shashirangana, H. Padmasiri, D. Meedeniya, et al. (2021), "License plate recognition using neural architecture search for edge devices", *International Journal of Intelligent System*, DOI: 10.1002/int.22471.

[16] NVIDIA (2023), *NVIDIA Jetson*, https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/, accessed 15 August 2023.

[17] N. Awalgaonkar, P. Bartakke, R. Chaugule (2021), "Automatic license plate recognition system using SSD", *2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA)*, pp.394-399, DOI: 10.1109/IRIA53009.2021.9588707.

[18] Raspberry Pi (2023), *Raspberry Pi*, https://www.raspberrypi.com/about/, accessed 15 August 2023.

[19] A.O. Agbeyangi, O.A. Alashiri, A.E. Otunuga (2020), "Automatic identification of vehicle plate number using Raspberry Pi", *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*, pp.1-4, DOI: 10.1109/ICMCECS47690.2020.246983.

[20] D.D. Nguyen, M.H. Le (2023), "Vietnam license plate segment datasets", *Kaggle*, https://www.kaggle.com/datasets/duydieunguyen/licenseplates, accessed 15 August 2023.

[21] X. Shao, C. Xu, J.H. Lim (2003), "Image mosaics base on homogeneous coordinates", *Institute for Infocomm Research*, 4pp.

[22] R. Padilla, S.L. Netto, E.A.B.D. Silva, "A survey on performance metrics for object-detection algorithms", *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp.237-242, DOI: 10.1109/IWSSIP48289.2020.9145130.