# Automatic License Plate Recognition Using OpenCV

Michael Kročka
*Faculty of Informatics and Information Technologies Slovak University of Technology in Bratislava*
Bratislava, Slovakia
xkrocka@stuba.sk

Pavle Dakić
*Faculty of Informatics and Information Technologies Slovak University of Technology in Bratislava*
Bratislava, Slovakia
pavle.dakic@stuba.sk

Valentino Vranić
*Faculty of Informatics and Information Technologies Slovak University of Technology in Bratislava*
Bratislava, Slovakia
vranic@stuba.sk

2022 12th International Conference on Advanced Computer Information Technologies (ACIT) | 978-1-6654-1050-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ACIT54803.2022.9913168

***Abstract*—Automatic license plate recognition is a technology that relies on optical character recognition methods used on images for reading license plate text. These techniques are, in general, used with image processing techniques, whether the system is composed of them entirely or in addition to deep learning techniques and neural network models. Such a system can be used by CCTV cameras, smart cameras, and other such devices, for tasks in areas like law enforcement, toll collection systems, or parking lot ticket systems. These systems can be used to capture images, locate license plates and recognize the text from the license plate. Depending on the region, post-processing can be conducted to further refine the recognized text based on a set of predefined rules, such as the position of letters and numbers in the license plate string. This work aims to create an automatic license plate recognition system with the use of image processing techniques, mainly from the OpenCV python library. The system will attempt to recognize license plate text from vehicles parked in parking lots. This can be useful for people, who do not remember the place where they parked their vehicle, or in other areas such as law enforcement. For large parking lots, there are generally camera systems already in place, in this case, the system should be able to work with multiple angles and views of the parking lot to get results with the highest accuracy ratings.**

***Index Terms*—ALPR - Automatic License Plate Recognition, OCR - Optical Character Recognition**

## I. Introduction

Overseeing car parks, highways, tollways, or just generally securing the safety of citizens, Automatic License Plate Recognition (ALPR) systems can be very helpful. These systems capture license plates and use optical recognition methods to discern the characters in the plate and turn them into text. In the context of parking lots, which this paper focuses on, these systems can be useful for when vehicles enter and/or leave car parks, to automatically open gates, which in turn allows for more fluid traffic in the area. It can also be used for finding vehicles with said license plate numbers in large, or hard-to-navigate parking lots, enabling customers to quickly navigate their way to their automobiles - again increasing the fluidity of the traffic. The goal of this work is to create a system that from a single occupied parking space image can accurately and in real-time detect and recognize the number plate text. To achieve this, we contemplated using deep learning techniques which are prevalent in the research area but ultimately opted for image processing methods utilizing mainly the OpenCV python library. The system will run on a Raspberry Pi Model 4B equipped with a camera module. The Optical Character Recognition (OCR) will be done using Tesseract OCR developed by Google to transform the license plate image into text.

## II. Research Methods

We describe the test data, methods, thought processes, ideas, libraries and technologies which were used during experiments.

### A. Data

The majority of our testing and development was conducted using the **500 Image of the Rear View**, which is a dataset for license plate detection, recognition, and automated storage [1]. It can be downloaded freely as it is open-source1. This dataset was generated as a part of a students project led by prof.dr.sc. Slobodan Ribarić [1]. The dataset contains 500 unique images of multiple vehicle types along with their license plates. The images vary in size, the license plates themselves vary in shape, color, as well as the lighting conditions, tilt angles, and obstructions. The dataset summary can be found in the table below. The vehicle types in the dataset are varied, mainly composed of personal automobiles of various sizes, but small vans, buses, and large semi-trucks are also present.

TABLE I. Statistics for 500 Image of the Rear View dataset

| Shape | | Color | | Tilt | | Clear | |
|---|---|---|---|---|---|---|---|
| Rectangle | Square | Black-White | Colored | Yes | No | Yes | No |
| 476 | 24 | 486 | 14 | 39 | 461 | 454 | 46 |

Square license plates are those whore width height ratio is close to 1 but not greater than 1.05. An example is given in the figure 1.


Fig. 1. Images of square-shaped license plate on the left and rectangle-shapedon the right. Source: [1]

Colored license plates are considered as those, whose back- ground is not white and the text is not black. An example is given in the figure 2.

[1]https://platerecognizer.com/number-plate-datasets/

Fig. 2. Images of colored license plate on the left and black-white on theright. Source: [1]

Tilted license plates are those, whose tilt angle is greater than 10 degrees. Such examples can be seen in the figure 3.

Fig. 3. Images of tilted license plate on the left and what is considered nottilted on the right. Source: [1]

Non-clear license plate images are those which are either:

1) Too dirty to be read.
2) The lighting conditions are poor, e.g. there is a shadow or a strip of strong ray of sunshine hindering the detection.
3) The license plate itself is deformed or defective.

Fig. 4. Images of license plate with strong shadows on the left and deformed license plate on the right. Source: [1]

For testing purposes, a hand-made CSV file containing x and y coordinates of the top-left and bottom-right points ofthe license plate bounding rectangles was constructed. The bounding rectangles themselves were based on the character contours, not the *vehicle plates* themselves. For clarification, an example is shown in the figure 5.

In addition to the **500 Image of the Rear View** dataset, experiments were also conducted using the **UFPR-ALPR** dataset [2], also open-source[2]. This dataset, however, did not prove to be well suited for the purpose of our work, as the images were taken from vehicles in traffic. This posed in itself many challenges, such as more extreme tilt angles, lowerimage quality, and pixel count, different vehicle types suchas motorbikes and mopeds.

[2]https://platerecognizer.com/number-plate-datasets/


Fig. 5. Example of Bounding rectangle used for constructing x and y coordinates csv file. Source: Own image.

Also, present in the dataset are license plates with a very peculiar color scheme native toBrazil, seen in figure 6, which we decided not to incorporate into our solution. As an afterword, during the latter half of February 2022 a new dataset was published under the name **RodoSol-ALPR** dataset, which was also developed by Laroca et. al. [3]. Very superficial experiments were conducted using this new dataset.


Fig. 6. Images of vehicles form the UFPR-ALPR dataset with red licenseplate colour scheme. Source: [2]

### *B. Methods*

The development was done using the Python programming language and Google Colab cloud service, as the hardware available at hand was not suitable for long-running tests. The majority of the program relies on the OpenCV python library and could be divided into multiple parts:

1) License plate localization
2) License plate cleaning
3) Character detection
4) Character recognition using PyTesseract OCR

The preprocessing of the image begins with loading the image, continues with finding the license plate, and ends with segmenting license plate characters. Numpy and OpenCV python libraries were used for most of the operations in the preprocessing pipeline. The images are

transformed to OpenCV UMat objects during runtime, a unified abstraction that allows functions to use an OpenCL-enabled GPU if it exists in the system and automatically switch over to CPU operations otherwise [3].

For the **license plate localization**, there were two approaches that were tested. The first is to focus on areas in the image where there is a high concentration of edges. This is backed by the logic that the license plate characters are easily distinguished as there is high contrast between the black (dark)character contour and white (light) background. The second approach is to focus on light areas of the image, this is based on the fact that the background of the license plate is of light color. We produce a light mask according to this logic and perform a bit-wise AND operation using a preprocessed image from the first approach. An overview of the two approaches can be seen in figure 7.

Figure reffig:get contours pipeline showcases the multiple steps of each of the two license plate detection pipelines. Basic image processing operations are used, such as gray-scale transformation, Gaussian Blurring, adaptive threshold, and open and close operation. At different times the operations are performed with different-sized kernels. The decisions and reasoning for the use of the aforementioned techniques willbe explained in the following paragraphs.

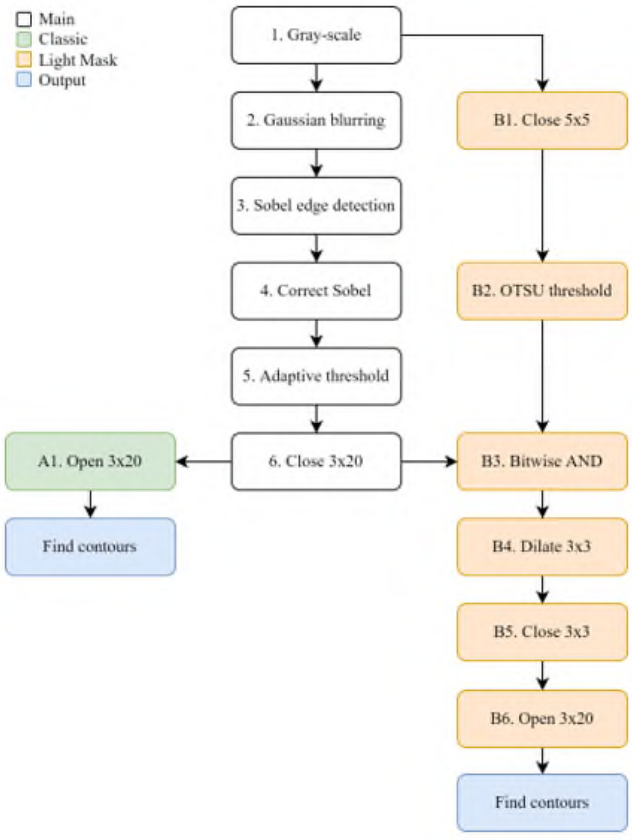


Fig. 7. The general pipeline of finding license plate contours. Source: Ownimage.

[3]https://docs.opencv.org/3.0-rc1/db/dfa/tutorial transition guide.html

Gaining speed in modern systems is in most cases basedon the use of descriptors that work on the principle of finding shades of gray in images. This, however, requires additional processing power when converting from color to black and white images. On the other hand, if hardware requirements were to be ignored, the degree of detection increases due to less complex data. After the conversion itself, an average or mean value is found for each of the regular red, blue, and green color channels. Most often, the emphasis is placed on the brightness in order to apply methods for generating a detectionmodel, that is maintained on a more accurate presentation of subjectively attractive information. The information obtained may relate to the reduction or enhancement of shadows, contrast, application of various filters and so on [4].

Another important technique used in modern systems is denoising. Denoising itself is a very general term, but in computer graphics and image preprocessing it is generally linked to commonly used techniques such as Gaussian Blurring, Bilateral Filtering, Median Blurring, or Averaging. Thesetechniques employ a filter that is applied to the input image and as a result, depending on the filter kernel, the image is lessnoisy. There are trade-offs to be made by using a specific typeof denoising algorithm, generally, Gaussian Blurring offers both adequate speed and result, while techniques such as Bilateral Filtering for example are more computationally heavybut keep edges sharp. Both gray-scale and color images can contain lots of noise or random variation in brightness or hue among pixels. The pixels in these images have a high variance,which just means there's lots of variation within groups of pixels. Because a photograph is two-dimensional, Gaussian Blurring uses two mathematical functions (one for the x-axis and one for the y) to make the 3rd function, also called a convolution [5], [6].

For the edge detection algorithm, we chose to work withthe Sobel operator as applies an average filter and is not as affected by noise. Other than we conducted experiments with the Roberts and Laplacian operators as well. The Roberts edgedetection algorithm can locate edges very accurately but does not detect noise in images. On the other hand, the Laplacian edge detection algorithm measure edges from all directions and can detect fine edges with high precision, but resultsin dual pixel edges when faced with noisy images. Lastly,the Sobel operator is directional by nature and will produce different results when applied to the x or the y, or both the axis [7].

The license plate cleaning segment of the system attempts to more clearly detect the license plate region by filtering out the background noise. This is done by enhancing the contrast in the image using Adaptive Histogram Equalization followed by bilateral filtering. At this point we opted for bilateral filtering as the image size is much smaller, regulated by the rules for license plate candidates, so the computational workload is not as heavy. The license plate contour candidate is ready for the threshold operation. We perform the adaptive threshold function from the OpenCV library, again opting for better results rather than speed as per the reason given before. If we wanted to, however, focus on speed, the OTSU

thresholdmethod could be used. The time difference between the two methods was not as impactful in our tests, so there was no reason to sacrifice accuracy here. The bounding rectangle of the cleaned contour is then tested against tighter ratio checks for both rectangular and square-shaped plates. At this point, the contour does not have to pass the ratio check and a perspective transformation is attempted.

The character detection part focuses on a set of rules, by which character candidates are filtered. At the start, the discerned contour candidates are sorted from left to right, in order to keep to the character order present on the license plate. Assuming that the license plate cleaning took effect, we conclude that the contour area of the character has to be larger than 1% of the total license plate image area, but also smaller than 20% of said area. The range may seem large, but it was constructed based on the wide range of image sizes, plate rotations, and other aspects present in the dataset. Furthermore, we test if the character contour candidate has an aspect ratio smaller than 1, as well as whether the height ofthe contour is bigger than the width.

For the Optical Character Recognition we chose to use PyTesseract OCR developed by Google, rather than try to trainan OCR neural network from scratch. In the long run, it would be beneficial not to take such shortcuts, but given the available time frame, we opted for this route. The PyTesseract can be configured by setting engine modes and page segmentation modes. For our purposes, we opted for the engine mode whichuses both the original Tesseract and the LSTM neural nets.As for the page segmentation mode, we preprocessed ourimages to be in the form of a single line of text, therefore we set the page segmentation to also await images in such a format. During our tests, there was a large contrast between images of black text on a white background and vice-versa with the former being much better overall. Other than that,we attempted to follow the Tesseract guideline to improve the quality of the recognition. The most important ones for uswere the image inverting and re-scaling, as Tesseract seems towork best on images with a DPI of at least 300.

## III. Related Work

Texture features depend heavily on the presence of characters in license plates. Grey-scale images are generally used to further amplify the jump between the character and back- ground texture. Approaches vary, from Sliding Cascade Windows (SCW), usage of Gabor filters, Discrete Fourier Trans- formations (DFT), and many more [8], [9]. The mentioned approaches have the ability to detect deformed license plates, but the drawback is that they are usually computationallyexpensive, mainly because of the presence of multiple edgesor texts in input images.

Color features can be used as edge identifiers since in somecountries a license plate color is region-specific. There are multiple approaches, but the main advantages of these methodsare robustness against tilted and otherwise deformed license plates. The difficulties sometimes outnumber the positives,because some of these methods are sensitive to light changes (RGB models), some are sensitive to noise (HLS models) and others encounter problems when the license plate number is prevalent in the input image [10].

Methods using character features as the defining characteristic of license plates are not preferred. With a ”binarized” image as input, the methods tend to be time-consuming and encounter difficulties when some other text is present [10].

Edge information is often utilized to extract license plates since they have a rectangular shape with a fixed aspect ratio. Research has shown that vertical edges are more robust than horizontal ones, as horizontal edges lead to errors considering car bumpers are also horizontal lines [11]. Hough transformation could be considered, it can detect straight lines inclinedup to 30°, but has a drawback of being time and memory consuming [12]. Edge-based detection of license plates is fast,but it is necessary for the edges to be uninterrupted.

Methods combining two or more features are called hybrid extraction methods [10]. In [13] a neural network is used, which scans the input image using a H W×window similar to the license plate dimensions to detect the edges and colors present in the window to decide whether the scanned region is a suitable candidate or not. Other works may work with different feature combinations, varying in the number of features applied and the way they are applied. All in all, these methods provide reliable results in multiple scenarios but are more computationally complex [10].

Isolated license plates may be tilted or have different lighting, which is why preprocessing is done before the segmentation methods are applied. The techniques for preprocessing will not be covered in this paper.

Segmentation using pixel connectivity is done by labeling the “binarized” input image of a license plate. These methods then analyze labeled pixels and may fit character templates or consider the aspect ratio [10], [14]. These methods are not robust against joined or malformed characters.

Projection profiles methods stem from the fact that the license plate characters and backgrounds have different colors,ergo they have opposite binary values. Some works project these vertically to extract the starting and ending point of characters and project the extracted characters horizontallyto extract each character separately [10]. Other works with color information instead of binarizing the input image. These methods are heavily dependent on the image quality, are sensitive to noise and the characters in the license plates must be known beforehand. The positive part of these methods is that they are not dependent on the spacing of the characters.

Prior knowledge of characters can be used to help the segmentation process. A binary image is scanned using a horizontal line to detect the starting and ending point of characters and when a ratio condition is satisfied, it is considered astarting point for the character [15]. Another method may beto resize the extracted license plate and to fit a template [10].

## IV. Formulation of Research Questions

This work is a part of a larger system that at first attempts to label parking spaces as occupied or empty using a pre-trained light-weight convolutional neural network. Afterward, using the steps described in section II-B, if the parking space is deemed to be occupied this part of the system will attemptto retrieve the license plate characters in text form. The system as a whole is meant to be run on devices with lower computational power, such as Raspberry Pi or others.

In our work environment, we do not have high-end computational equipment readily available, so we opted against machine learning and deep learning techniques and instead focused on ”easy to implement” solutions using image processing via the OpenCV library. For in-person testing, we have prepared a smart camera consisting of Raspberry Pi Model 4Bequipped with a high-definition camera together with a wide- angle camera lens. The Pi runs on an external 10000 mAh battery, but can easily be switched to be plug powered.

Fig. 8. Top view of the RPi setup with the case closed. Source: Own image.

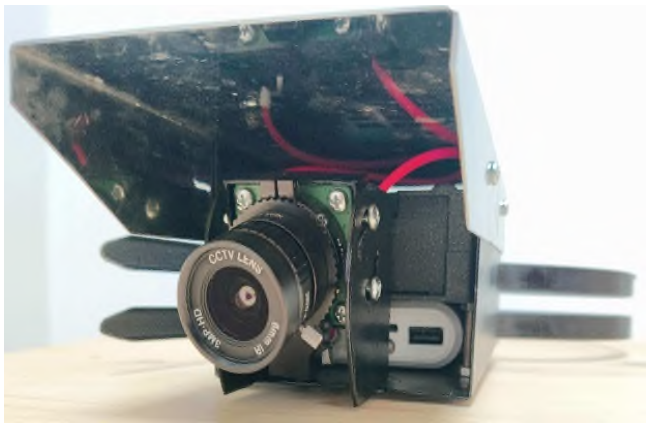

Fig. 9. Front view of the RPi setup with the case closed. Source: Own image.

As such, it was important for us to create a license plate detection and recognition pipeline that was not resource-heavy and could be run on setups such as a distributed network of smart cameras. One such concern was what if multiple cameras worked together to provide the best possible results available. That way each camera could be restricted to a number of parking spaces of which it has a clear and unimpeded view - providing reliable results. For parking spaces that are onthe edge of the view of such a camera, cross-referencing the output from another camera could be viable.

In modern parking lots, there often is a CCTV system in place, so the logical solution would be to have a central server that would take all the image camera input and perform the heavy-duty object detection and character recognition. Our approach is not suited for such environments, as it focuseson speed rather than accuracy.

## V. Analysis and Discussion

In the initial search for license plate contour candidates,we opted for algorithms and methods that were focused on computational speed rather than accuracy. After finding promising candidates we employed techniques that were more computationally heavy but provided better results. This was due to the scale of the candidates and the restrictions placed upon them. The major bottleneck in the speed of our program is the execution of PyTesseract OCR image-to-text functionality. In an ideal situation, we would beforehand know the pattern that the license plates follow, therefore we could filter out cases where such a pattern is not applicable. It would also help us in getting better accuracy results for OCR, as we wouldknow in which positions letters and numbers would be.

In the context of 500 Image of the Rear View dataset using the manually created CSV file containing x and y coordinates of top-left and bottom-right, we compute the accuracy percentage for locating license plates as well as to measure the time. OCR Sum Score is calculated by the similarity between the predicted text and the real text, if they are equal, then the score would be equal to 1. In our experiments, the maximum OCR Sum Score would be 500.

TABLE II. Accuracy and speed statistics for 500 Image of the Rear View dataset computed using standard Google Colab environment

| | **Combined** | **Edge focused** | **Light mask** |
|---|---|---|---|
| LP detected | 481 (96.2%) | 468 (93.6%) | 433 (86.6%) |
| OCR passed | 450 (90.0%) | 422 (84.4%) | 405 (81.1%) |
| Time per image | 423.8ms | 321.0ms | 325.0ms |
| Time OCR per image | 364.8ms | 268.3ms | 274.8ms |
| OCR Sum Score | 418.58 | 390.79 | 378.15 |

One thing we noticed during testing is that our algorithm is not well suited for colored license plates, especially thosewhere the contrast between the character and the backgroundcolor is not high enough. For these cases, we have concludedthat alterations to the adaptive threshold value would fix theproblem. Other options include the usage of algorithms for theincrease of contrast, e.g. variations of histogram equalization.

As can be seen from the table, the majority of the timeis taken up by the Tesseract OCR. This could be further optimized by restricting the conditions under which the OCR process would take place. One clear candidate would be the aforementioned pattern matching, given that we knew the general pattern that the license plate follows. Other than that, if we could ensure the license plate is always near the image center, we could restrict our search to such an area.

## VI. Conclusion

In this paper, we outlined the general structure of our Automatic License Plate Detection algorithm. We gave insight intothe two main ideas for our license plate detection pipelines, one focusing mainly on areas of the image with a concentrationof edges, while the other focuses on light image areas. The reasons and motivation behind the selection of certain image processing techniques such as gray-scale transformation, Gaussian Blurring, Adaptive and OTSU threshold operations, and perspective transformations were outlined. Experiments were concluded on the 500 Image of the Rear View dataset, along with glaring points to be improved upon.

In our future work, we would like to better define conditions on when to perform Optical Character Recognition, as this is the main speed bottleneck of the program. We would also like to increase the functionality of our system to be able to accurately work with color license plates. We look forward to testing our system on other datasets and future improvements.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgment

The work reported here was supported by the Scientific Grant Agency of the Slovak Republic (VEGA) under grant No. VG 1/0759/19 and the Operational Programme Integrated Infrastructure for the project: Advancing University Capacity and Competence in Research, Development, and Innovation (ACCORD), ITMS code 313021X329, co-funded by the European Regional Development Fund (ERDF).

## References

[1] S. Riaric, “License plate detection, recognition, and automated storage,” *Univ. Zagreb, Zagreb, Croatia, Rep*, 2003.

[2] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves,
W. R. Schwartz, and D. Menotti, “A robust real-time automatic license plate recognition based on the yolo detector,” in *2018 international jointconference on neural networks (ijcnn)*. IEEE, 2018, pp. 1–10.

[3] R. Laroca, E. V. Cardoso, D. R. Lucio, V. Estevam, and D. Menotti, “On the cross-dataset generalization for license plate recognition,” *arXivpreprint arXiv:2201.00267*, 2022.

[4] C. Kanan and G. W. Cottrell, “Color-to-grayscale: does the method matter in image recognition?” *PloS one*, vol. 7, no. 1, p. e29740, 2012.

[5] S. S. Al-Amri, N. V. Kalyankar *et al.*, “Deblured gaussian blurred images,” *arXiv preprint arXiv:1004.4448*, 2010.

[6] L. Shapiro and G. Stockman, “C:” computer vision”, page 137, 150. prentice hall,” 2001.

[7] P. P. Acharjya, R. Das, and D. Ghoshal, “Study and comparison of different edge detectors for image segmentation,” *Global Journal of Computer Science and Technology*, 2012.

[8] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and
E. Kayafas, “A license plate-recognition algorithm for intelligent transportation system applications,” *IEEE Transactions on Intelligent transportation systems*, vol. 7, no. 3, pp. 377–392, 2006.

[9] H. Caner, H. S. Gecim, and A. Z. Alkar, “Efficient embedded neural- network-based license plate recognition system,” *IEEE Transactions on Vehicular Technology*, vol. 57, no. 5, pp. 2675–2683, 2008.

[10] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, “Automatic license plate recognition (alpr): A state-of-the-art review,” *IEEE Transactions on circuits and systems for video technology*, vol. 23, no. 2, pp. 311– 325, 2012.

[11] S.-Z. Wang and H.-J. Lee, “Detection and recognition of license plate characters with different appearances,” in *Proceedings of the 2003 IEEEInternational Conference on Intelligent Transportation Systems*, vol. 2. IEEE, 2003, pp. 979–984.

[12] T. D. Duan, D. A. Duc, and T. L. H. Du, “Combining hough transform and contour algorithm for detecting vehicles’ license-plates,” in *Pro- ceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004.* IEEE, 2004, pp. 747–750.

[13] J.-F. Xu, S.-F. Li, and M.-S. Yu, “Car license plate extraction using colorand edge information,” in *Proceedings of 2004 International Conferenceon Machine Learning and Cybernetics (IEEE Cat. No. 04EX826)*, vol. 6.IEEE, 2004, pp. 3904–3907.

[14] K. Kanayama, Y. Fujikawa, K. Fujimoto, and M. Horino, “Developmentof vehicle-license number recognition system using real-time image processing and its application to travel-time measurement,” in *[1991 Proceedings] 41st IEEE Vehicular Technology Conference*. IEEE, 1991,pp. 798–804.

[15] C. Busch, R. Domer, C. Freytag, and H. Ziegler, “Feature based recognition of traffic video streams for online route tracing,” in *VTC’98. 48th IEEE Vehicular Technology Conference. Pathway to Global Wireless Revolution (Cat. No. 98CH36151)*, vol. 3. IEEE, 1998, pp. 1790–1794.