International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

SINGLE LINE LICENSE PLATE DETECTION USING OPENCV AND TESSERACT

Tushar Goel[1], Dr. K.C. Tripathi[2], Dr. M.L. Sharma[3]

[1]Student, Dept. of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, India
[2]Associate Professor, Dept. of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, India
[3]Head of Department, Dept. of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, India

---

Abstract – *License plate detection is an image processing technology that uses a license (number) plate for vehicle identification. The objective is to design and implement an efficient vehicle identification system that identifies the vehicle using the vehicle's license plate. The system can be implemented on the entrance of parking lots, toll booths, or any private premises like college, etc. to keep the records of ongoing and outgoing vehicles. It can be used to allow access to only permitted vehicles inside the premises. The developed system first captures the image of the vehicle's front, then detects the license plate and then reads the license plate. The vehicle license plate is extracted using the image processing of the image. Optical character recognition (OCR) is used for character recognition. The system is implemented using OpenCV and its performance is tested on various images. It is observed that the developed system successfully detects and recognizes the vehicle license plate*

Key Words: **OpenCV, license plate recognition, image processing, optical character recognition, ALPR**.

1. INTRODUCTION

License plate detection also considered and automatic license plate recognition (ALPR) has various applications such as parking lot management, stolen vehicle identification, traffic flow monitoring, electronic toll collection, etc. This topic has been extensively researched by researchers worldwide to improve the performance of the ALPR in real-world scenarios [1]. License-plate detection or number plate detection uses optical character recognition (OCR) on the license plate image to recognize and extract the characters of a vehicle number plate. It is usually aided by cameras designed specifically for such a task, since the license-plate recognition may be especially difficult under poor images [2]. The use of automated and intelligent license plate recognition system is needed in today's real-time processing age. The growing number of vehicles demands a reliable and robust system which along with license plate detection from a vehicle provides the text output of the characters printed on the plate. Although a number of systems have already been devised for this function, only a handful of them has been able to implement Optical Character Recognition [3]. The system proposed in this paper has been implemented using the OpenCV library using Python language is used for image processing and using PyTesseract for optical character recognition (OCR). Pytessaract is used for text extraction from the processed license plate image. This system has been implemented keeping in mind that the characters are in a single line. After the image processing step and the plate detection step is employed the output text is formatted by filtering out semicolons, commas, colons, apostrophes, and other such special characters using ASCII filtering as these characters are not part of any standard license plate [4].

2. METHODOLOGY

License plate of the vehicle is detected using various features of image processing library openCV and recognizing the text on the license plate using python tool named as tesseract. To recognize the license plate we are using the fact that License plate of any vehicle has rectangular shape. So, after all the processing of an image we will find the contour having four points inside the list and consider it as the License Plate of the vehicle.

1.1 Import Libraries and Image

To implement the project first various python tools and libraries are imported. I had imported four libraries OpenCV for image processing, Numpy for mathematics, Matplotlib for plotting an image and Pytesseract for optical character recognition (OCR).
After the libraries are imported I import the image using its path and store the image in variable named as image.

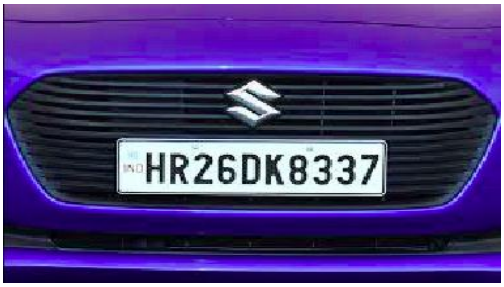

Fig -1: Original Image

1.2 Preprocessing

A colored image is an image in which each pixel is specified by three values one each for the red, blue, and green components of the pixel scalar. M*N*3 array of class. To store a single color pixel of an RGB color image we will need m*n*3 bits, but when we convert an RGB image to a grayscale image, only m*n bits are required for storage of a single-pixel of an image. So we will need 33 percent memory for the storage of grayscale images than to store an RGB image. Grayscale images are much easier to work within a variety of tasks like In many morphological operations and image segmentation problems, it is easier to work with the single-layered image (Grayscale image) than a three-layered image (RGB color image). It is also easier to distinguish features of an image when we deal with a single-layered [5].

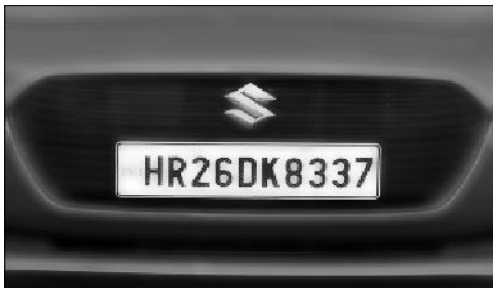

Fig-2: Grayscale Image

After gray scaling we will blur the gray image to reduce the background noise. Image blurring is done by passing an image with the low-pass filter kernel. It is very useful for removing noise. It removes high-frequency content from the image. So edges are blurred in this operation but there are also blurring techniques that don't blur the edges. There are different blurring methods that can be used to blur the gray image [6]. Averaging (first method) is done by convolving an image with a normalized box filter. This method takes an average of all the pixels under the kernel area and assigns the central element. In the Gaussian Blurring method (second method), instead of a box filter, a Gaussian kernel is used. We specify the height and width of the kernel which should be odd and positive. We also specify the standard deviation in the Y and X directions, sigma X, and sigma Y respectively. Median Blurring (third method) takes the median of all the pixels under the kernel area and the central element is assigned with this median value. This is highly effective against pepper-and-salt noise in the image. Its kernel size should be an odd and positive integer. Bilateral Filtering (fourth method) is highly effective in noise removal and keeping edges sharp. This operation is slower as compared to other filters. Bilateral filtering takes a Gaussian filter, but one more Gaussian filter which is a function of pixel difference so it does not affect the edges [7]. I had used the bilateral filter to blur the image because it actually preserves all strength, it removes noise quite well and strengthens the edges in the image when we deal with a single-layered image.

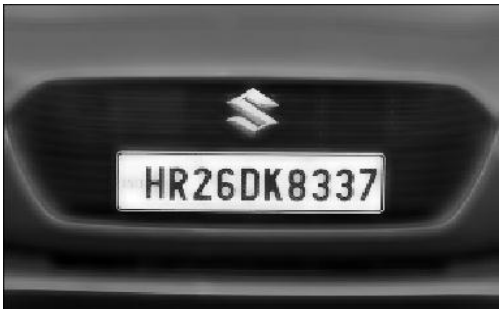

Fig-3: Blurred Image

After blurring we will do edge detection. It is a very important part of computer vision, especially when we are dealing with contours. Edges are defined as sudden changes in an image. They can encode just as much information as pixels. Edges are also defined as the boundaries of the images. There are three main types of Edge Detection. Sobel Edge Detection (first method) is a way to avoid the gradient calculated about an interpolated point between the pixels which uses 3 x 3 neighborhoods for the calculations of the gradient. It finds vertical or horizontal edges. Laplacian Edge Detection (Second method) builds a morphing algorithm that operates on features extracted from target images. It is a good method to find the edges in the target images. Canny Edge Detection (Third method) follows the series of steps and is a very powerful edge detection method. First it smoothens an image with the Gaussian filter. Then it computes the gradient magnitude and orientation using finite-difference approximations for the partial derivatives. Then it applies non-maxima suppression to the gradient magnitude. After this in the next step uses the double threshold algorithm to link and detect edges. Canny edge detector approximates the operator that optimizes the product of localization and signal-to-noise ratio. It is generally the first derivative of a Gaussian [7][8]. We will use the Canny edge detection to extract the edges from the blurred image because of its optimal result, well-defined edges, and accurate detection.

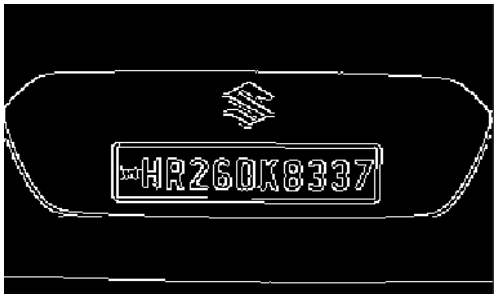

Fig-4: Edged Image

After finding edges we will find the contours from and edged image. Contours are the continuous curves or lines that cover or bound the full boundary of the object in the image. Contours play a very important role in object detection shape identification. There are mainly two important types of contours Retrieval Modes. The first one is the cv2.RETER_LIST which retrieves all the contours from an image and second is cv2.RETER_EXTERNAL which retrieves external or outer contours from an image. There are two types of Approximation Methods. The first method is the cv2.CHAIN_APPROX_NONE stores all the boundary points. But we don't necessarily need all bounding points. If the points form a straight line, we only need the start and ending points of that line. The second method is the cv2.CHAIN_APPROX_SIMPLE instead only provides these start and endpoints of bounding contours, thus resulting in much more efficient storage of contour information.

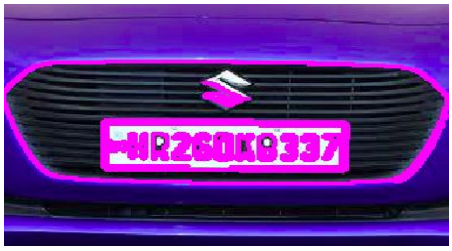

Fig-5: Image with Contours

After finding contours we will sort the contours. Sorting contours is quite useful when doing image processing. We will sort contours by area which will help us to eliminate some small and useless contours made by noise and extract the large contours which contain the number plate of a vehicle. We will take the top 10 contours to find our number plate as we are only using the frontal image of the car and we can expect the area of number plate covers most of the region in an image.

1.3 Detecting Plate

After we sort the contours we will now take a variable plate and store a value none in the variable recognizing that we did not find number plate till now. Now we iterate through all the contours we get after sorting from the largest to the smallest having our number plate in there so we should be able to segment it out. Now to that, we will look through all the contours and going to calculate the perimeter for the each contour. Then we will use cv2.approxPolyDP() function to count the number of sides [9]. The cv2.approxPolyDP() takes three parameters. First one is the individual contour which we wish to approximate. Second parameter is Approximation Accuracy Important parameter is determining the accuracy of the approximation. Small values give precise- approximations, large values give more generic approximation. A good rule of thumb is less than 5% of the contour perimeter. Third parameter is a Boolean value that states whether the approximate contour should be open or closed. I had used contour approximation and it approximate a contour shape to another shape with less number of that is dependent on the position I specify so the 0.02 is the precision that worked. After that we will compare if edges count is equal to 4 so we found our number plate.

After that we will find the coordinates of the rectangle formed using cv2.boundingRect(c) and store the one coordinate in x, y and store width and height of the contour in another. After that we put the image of detected rectangle in the plate variable.

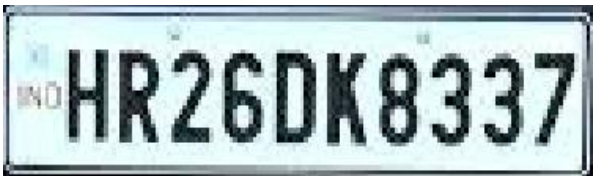

Fig-6: Detected License Plate

1.4 Text Recognition

After detecting the license plate of the vehicle we will recognize the characters on the license plate using tesseract. Python-tesseract is an (OCR) optical character recognition tool for python. That is, it will recognize and read|| the text embedded in images. It is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Leptonica imaging and Pillow libraries, including png, jpeg, gif, BMP, tiff, and others [10]. If Python-tesseract is used as a script it will print the recognized text instead of writing it to a file. Optical character recognition (OCR) is a conversion of printed text images or handwritten text scanned copy, into editable text for further processing. This technology gives an ability to the machine to recognize the text automatically. It is like a combination of the mind and eyes of the human body. An eye can only view the text from an image but the brain actually processes as well as interprets that extracted text read by eye.

```
print(text)
```

```
HR26DK8337
```

Fig-7: Output Text

3. SHORTCOMINGS

Although paper detects the license plate efficiently but method can be further improved using deep learning models and algorithms.

This method has been implemented keeping in mind that the characters are in a single line but there are number plates which contains more than one line.

Implemented method can give errors when an image contains multiple license plates or license plate occupying very small area in an image.

4. CONCLUSIONS & FUTURE SCOPE

Thus, an efficient and accurate, less time consuming method has been devised through this paper which can help in extraction and detection of license plate of vehicle. It promises to be less prone to errors if implemented in suitable conditions.

Further improvements can be done by using more advanced deep learning algorithms so it can work in every possible condition and can be implements in real-time monitoring, multiple license plate detection at a time, etc.

REFERENCES

[1] C. Henry, S. Y. Ahn and S. Lee, "Multinational License Plate Recognition Using Generalized Character Sequence Detection," in IEEE Access, vol. 8, pp. 35185-35199, 2020.
[2] H. Seibel, S. Goldenstein and A. Rocha, "Eyes on the Target: Super-Resolution and License-Plate Recognition in Low-Quality Surveillance Videos," in IEEE Access, vol. 5, pp. 20020-20035, 2017.
[3] P. S. Sharma, P. K. Roy, N. Ahmad, J. Ahuja and N. Kumar, "Localisation of License Plate and Character Recognition Using Haar Cascade," 2019 6th International Conference on Computing for Sustainable Global Development, New Delhi, India, 2019.
[4] R. R. Palekar, S. U. Parab, D. P. Parikh and V. N. Kamble, "Real time license plate detection using openCV and tesseract," 2017 International Conference on Communication and Signal Processing (ICCSP), Chennai, 2017.
[5] J. Chong, C. Tianhua and J. Linhao, "License Plate Recognition Based on Edge Detection Algorithm," 2013 Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Beijing, 2013.
[6] R. Huang, M. Fan, Y. Xing and Y. Zou, "Image Blur Classification and Unintentional Blur Removal," in IEEE Access, vol. 7, pp. 106327-106335, 2019.
[7] C. Xiong, L. Chen and Y. Pang, "An Adaptive Bilateral Filtering Algorithm and its Application in Edge Detection," 2010 International Conference on Measuring Technology and Mechatronics Automation, Changsha City, 2010.
[8] M. Kalbasi and H. Nikmehr, "Noise-Robust, Reconfigurable Canny Edge Detection and its Hardware Realization," in IEEE Access, vol. 8, pp. 39934-39945, 2020.
[9] R. Baran and A. Kleszcz, "The efficient spatial methods of contour approximation," 2014 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, 2014.
[10] K. Deb, M. I. Khan, M. R. Alam and K. Jo, "Optical recognition of vehicle license plates," Proceedings of 2011 6th International Forum on Strategic Technology, Harbin, Heilongjiang, 2011.