International Journal of Communication and Information Technology

E-ISSN: 2707-6628
P-ISSN: 2707-661X
www.computersciencejournals.com/ijcit
IJCIT 2025; 6(1): 18-23
Received: 06-11-2024
Accepted: 08-12-2024

Chandra Sekhar Sanaboina
Assistant Professor, Department of Computer Science and Engineering, University College of Engineering Kakinada, JNTUK-Kakinada, Andhra Pradesh, India

Corresponding Author:
Chandra Sekhar Sanaboina
Assistant Professor, Department of Computer Science and Engineering, University College of Engineering Kakinada, JNTUK-Kakinada, Andhra Pradesh, India

An innovative and cost effective IOT system for character recognition of vehicle license plates

Chandra Sekhar Sanaboina

DOI: https://doi.org/10.33545/2707661X.2025.v6.i1a.106

Abstract
Tracking the stolen vehicles, parking vehicle identification and parking the vehicles in the restricted areas is a big challenge nowadays. The main way of addressing these challenges is to recognize them through the number of plates embedded with technology. The task is therefore to recognize the position and the character on the number plate of the car. The current research article proposed an innovative cost-effective dashboard to identify the number of plates by capturing the image using a camera that is plugged with Raspberry Pi. The character recognition on the number plate is done by using K-Nearest Neighbors (KNN) algorithm. The images of the number plate are captured using raspberry pi cam and the numbers from the plate are extracted using Open CV. These identified number of plates will be trained using machine learning (ML) models. Based on the mean and variance the Euclidian distance will be calculated for training purposes. After training the large available datasets with all the unique number plates the trained data will be in the form of some encrypted format. So by inputting the new image (number plate) the features will be extracted and based on the calculation of the distance the rank will be allocated with the new plate and gets the nearest plate with character recognition the number will be extracted.

Keywords: KNN, Raspberry Pi, feature extraction, machine learning, variance

1. Introduction
The proposed system is an automated system that automatically identifies characters on the number plates. Various applications of the proposed system will be at the toll plaza, parking areas, data collection from the highway for crime prevention, etc. This system follows three major steps:

- Number plate identification
- Segmenting the characters with the boundary with Region of Interest (ROI).
- Character recognition.

Many challenges need to be solved to create the successful number plate recognition. To pinpoint some of them are low image quality, the shape of the image, noisy background and these parameters will take time to process the image to identify the characters. The need for identification is to prevent crime, parking fees, and toll collection and border control. Basic step is to extract the features of the vehicle image. And these features can be various and most prominent features are nothing but mean and variance. Once raspberry identifies the image and internally the image will be processed inside the raspberry itself from the open CV API. The first foremost step is to identify the border of the vehicle with the ROI using open CV in raspberry. This is called as canny edge detection in image processing and the back ground will be extracted and Region of Interest (ROI) will be stored for processing. Once the edge detection using open CV api the image will be marked with edge and inside the raspberry and edge detection will be stored as a temporary storage and the features will be extracted and all the features will be appended to vector table.

2. Related Work
There are several number plate/number detection that have been used before like edge extraction, morphological operations [1], vector quantization [5] and grayscale classification [2]. Due to ambient light conditions, interference characters, and other problems, its too difficult to detect the number plate/number in complex situations.

Some previous number plate recognition system will not work under some conditions like some strange backgrounds [3]. In previous works some research works have been under some complex conditions with the proposal of statistical features of number plate templates [4]. After that statistical features followed by ROI (region of interest), general number plate templates were applied to compare ROI. In so many cases, general number plate templates are very much difficult to be constructed. These kind of flows works on a fixed attributes. So the application of approach is restricted. The past approaches used character boundaries [7] as basic units of number plates, which will make the approach quite complex to check the point and illumination. This approach hardly can highlight characters overlapping from original number plates, and more over the using color edge and fuzzy disciplines has been proposed [8]. But this can be with certain limited colors with Open CV.

Vertical edge removal with open CV [6] is one of the most important identification measures because it requires the entire scheme to detect the plate perfectly. And map of edge has been used for vastly reduction of the complexity and it retains the main stricture which presents in the original image for the open CV has rich API on raspberry. So vertical edge operations [9] with raspberry was used for many years. The approaches provided use one-way Sobel operator to remove the labeled vertical edges. More about some unwanted information like horizontal edges are held in some vertical edge map as the open CV API parameters take marked/rounded picture horizontal and vertical borders [10].

3. Technologies used

1. Raspberry Pi

It is the controller of a series of raspberry board-based computers designed to explain fundamental computer science in education and contemporary nations.

The organization behind the Raspberry pi is basically with 2 arms. The first one with two models were developed by the Raspberry Pi foundation and later Pi model of B was released, so the foundation set up Raspberry Pi trade several versions of the Raspberry pi hardware feature modifications in memory ability and peripheral device support.

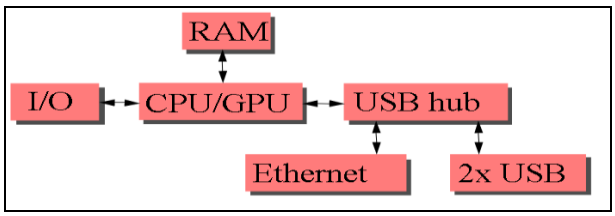


Fig 1: Block Diagram of Raspberry Pi

Figure 1 provides the Model B and B+ description, Model A, A+ and pie zero are comparable, but with no additional USB port. The USB port is attached / connected directly to the system on a chip call SOC in Model A, A+ and Pi zero. The USB / Ethernet module, which includes a five-slot USB hub with four ports, is accessible on the Pi 1 Model B+ and latest modules, while the Pi 1 Model B offers only two ports. The USB port is also directly linked to SOC on Pi Zero, but it utilizes micro USB (OTG0) as a port. Unlike other kinds of Pi, the 40pin GPIO connector is omitted on the Pi Zero with appropriate solder able holes only in the pin places. Pi zero WH remedies this.

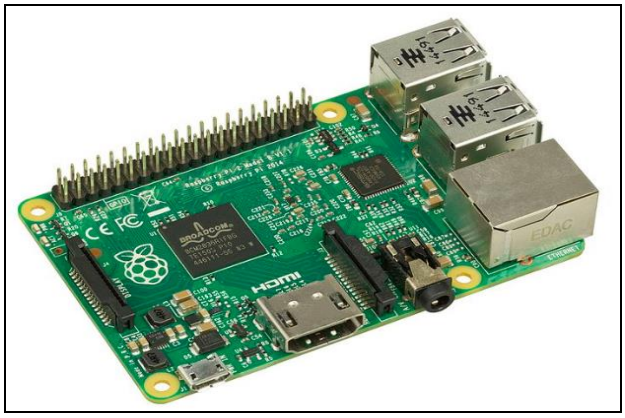


Fig 2: Raspberry Pi 3B

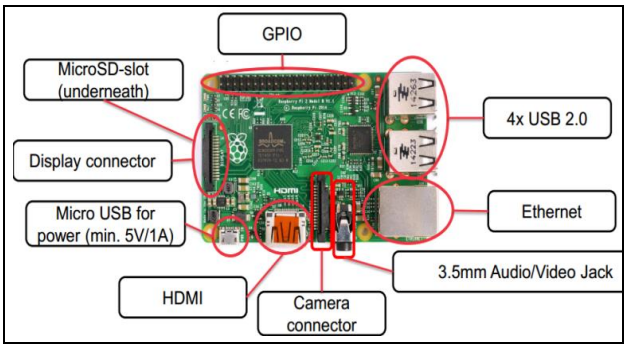


Fig 3: Raspberry Pi 3B internal components

Figure 2 depicts the Raspberry Pi3B component and Figure 3 depicts Raspberry Pi 3B internal components such as a 4-core ARM Cortex-A7 CPU of 32-bit 900 MHz.

2. Image processing with open CV on raspberry Pi

2.1 Open CV

Open CV (Open Source Computer Vision Library) is an API library for open source computer vision and machine learning. Open CV was constructed to provide a prevalent computer vision infrastructure and use machine perception in business products. Open CV makes it simple for businesses to use and customize the code. Figure 4 gives the broad overview of character recognition.

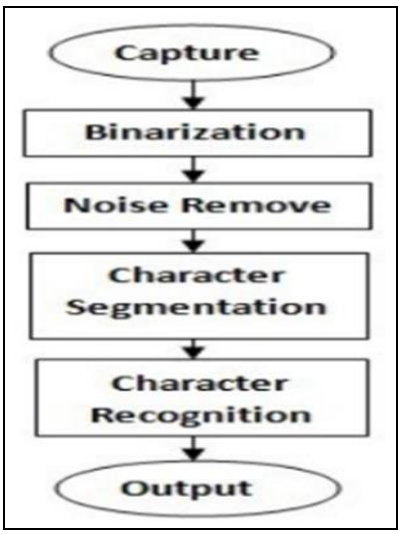


Fig 4: Broad overview of character recognition

2.2 Terminology involved in number plate recognition

2.2.1 Number Plate detection

The fore most step is to detect the license plate from the vehicle and in this work we have used the cars. This process will contour option of Open CV to detect the rectangular (vertical and horizontally) as objects to find the Region of Interest (ROI), (i.e., number plate). This work increased the accuracy based on the size of the exact size and color of the number plate. Basically the detection approach is the trained based on the KNN approach with respect to position of the camera and type of number plate used in the specific country/region. Some times OpenCv in this work gets tricker if the image from video does not even have a car, in that specific case this work can be extended to detect the vehicle (car) and then number plate.

2.2.2 Character wise segmentation

Once this approach detects the number plate from raspberry with cam and internally OpenCv Api crops and temperorly saved as a fresh image. Next time this can be repeated again using OpenCV knn libraries.

2.2.3 Character recognition with ROI: Now the stored and cropped image that saved on raspberry will be input to OpenCV API with knn library to obtain the charecters (numbers/alphabets) placed on it. By using OCR (Optical Character Recognition) library on raspberry can detect the number.

2.2.4 Optical Character Recognition (OCR): The main capacity of the machines to use a camera to look on the real world and to integrate the data from that it would be having the huge influence on their applications. This process will learn to identify the images by reading the characters present on it. This is known as OCR.

2.3 Steps involved in number plate recognition using raspberry pi

The flow of this entire process of license plate recognition is depicted in Figure 5.

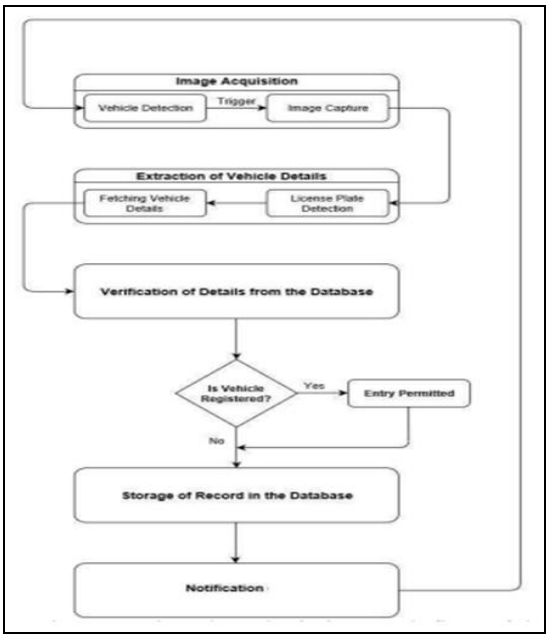


Fig 5: Steps involved in number plate recognition using raspberry PI

Step 1-Importing necessary libraries and capturing the image

OpenCV of the python library consists of an API which can detect License/Number plate of a vehicle. The code is run on the Raspberry pi component. The vehicle picture which is captured using the camera module of Raspberry Pi as show in Figure 6 is used for the character segmentation and identification also.

Fig 6: Captured image from Raspberry Pi

Step 2-Resizing the images

Then with OpenCV api the image will be resized to desired size and converts to grayscale and stored on the raspberry temporary image. Resizing will be helpful with the bigger images to unwanted parts of elimination and this makes sure that all the rectangular objects still remains on the gray scaled resized image.

Step 3-Conversion to gray scale images

Conversion of the gray scale images is common in image processing which speeds up the following steps in the process. The output of step 2 and step 3 is shown in Figure 7.

Fig 7: Resized and gray scale imag

Step 4-Removal of noise from the image: Every image will be having valuable and unwanted information, in this case only number/licence plate and rest are noise based portions. By using OpenCV KNN's bilateral fiter with blurring will erase the unwanted portions from the image.

The following instruction does this part

Gray=cv2. Bilateral Filter (Gray, 11, 17, 17)

The syntax of the target image will take four parameters
1. Source_image (gray)
2. Diameter of pixel (11)
3. SigmaColor (17)
4. SigmaSpace (17)

SigmaColor basically will be used to make it blur to extract the more of the background portion and the parameters has to be in proper so that wanted portions should not get blurred. Denoised output image of this step is shown in the Figure 8.

Fig 8: Denoised image

Step 5-Edge Detection: Followed by this is identifying the Region of Interest (ROI). This process can perform the edge detection. Many procedures are available to perform this action but the most popular with more accuracy is canny edge method from OpenCV. The following instruction on raspberry can perform this process.

Edged = cv2.canny (Gray, 30, 200)

The main syntax will use following parameters
1. Source image
2. Source preprocessed image
3. Threshold Value 2

Fig 9: Edge detected image

Step 6-Finding out the contours of image using OpenCV: Based on the image segmentation which is the process by which API can make portions of the images into more different regions. Whereas the contours are the continuous orthodoxed lines and curves which can cover the full boundary of the ROI objects in an image. Here this process of algorithm will use image segmentation and methods called as contours to extract the part by part of the image.
Main Contours very much useful for the following parameters.
1. Object detection/identification
2. Shape analysation

And these are very much broad field in application from real world image analysis to contour portion of images.

The following is the process of contour detection and output. Figure 10, Figure 11, Figure 12 and Figure 13 indicates input image for contour detection, canny edge for input image, Canny edge after contouring and Contours to input image respectively.

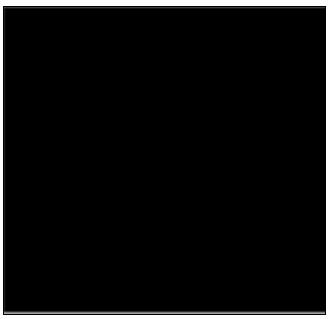


Fig 10: Input image for contour detection

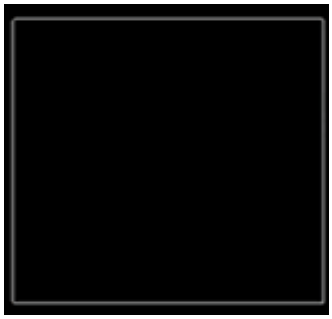


Fig 11: Canny edge for input image

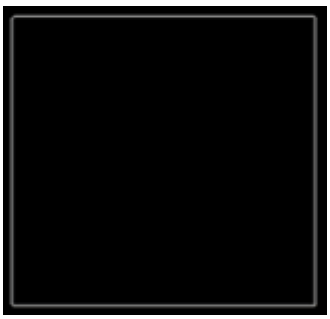


Fig 12: Canny edge after contouring

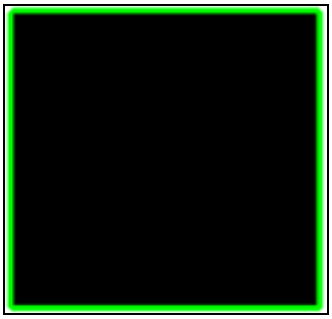


Fig 13: Contours to input image

Step 7-Sorting and considering contours within the threshold:
Once the contours have been identified the sorting will be done using KNN from big to small and consider the only first K-value result with threshold of 10 and ignoring others. In this image the contour could be anything that is having the closed and nearest surface but of all the obtained output results the license plate number will also be there since it is

also considered as closed surface.

Step 8-Filtration of number plate from the obtained contours

For filtration of the number plate with in the obtained results, this process has to be checked with rectangular portions with contour with four sides with closed Figure since the number plate would be rectangular shape.

```
# loop over our contours
for c in cnts:
                # approximate the contour
                peri = cv2.arcLength(c, True)
                approx = cv2.approxPolyDP(c, 0.018 * peri, True)
                # if our approximated contour has four points, then
                # we can assume that we have found our screen
                if len(approx) == 4:
                      screenCnt = approx
                      break
```

The value 0.018 is a practical and experimental value and it can be varied with the bast fit which can take to next level with machine learning to train based on the vehicle(car) images and then can be used with right fit value with knn approach. The correct contour will set in this approach with screen Cnt parameter of rectangular box around the detected object with in the portion of number plate properly.

Fig 14: Identification of number plate from input image

Step 9-Identification of number plate in the input image

The next process from the above Figure which was identified with number plate the remaining information is pretty much unwanted in this flow. So this process of OpenCV will proceed with masking with KNN threshold mask technique where the number plate is present. It is shown in Figure 14.

```
# Masking the part other than the number plate
mask = np.zeros(gray.shape,np.uint8)
new_image = cv2.drawContours(mask,[screenCnt],0,255,-1,)
new_image = cv2.bitwise_and(img,img,mask=mask)
```

Step 10-Number plate identification from the entire image

Figure 15 separates the number plate from the entire input image.

Fig 15: Identification of number plate

Step 11-Character segmentation

The next process is character segmentation to identify each and every character separately. This makes use of cropping technique and saving as new image on Raspberry Pi. This approach of Raspberry Number plate recognition is to actually reading the previous output and to do segmentation by using OpenCV pytesseract to read the characters from the image.

Step 12-Mathematical interpretation for converting RGB image to HSV image

The HSV scheme (hue, saturation, intensity) is a commonly used image processing RGB space that is appropriate for human vision. This is influenced by the reality that the yellow hue is less color insensitive than the other colors (such as the blue), we use HSV color space to find the Indian license plates.

Conversion of RGB color to HSV is done as follows:

$$\begin{cases} V = max(R,G,B) \\ S = \begin{cases} [V - min(R,G,B)] \times 255/V & V \neq 0 \\ 0 \quad \text{otherwise} \end{cases} \\ H = \begin{cases} (G-B) \times 60/S & V = R \\ 180 + (B-R) \times 60/S & V = G \\ 240 + (R-G) \times 60/S & V = B \end{cases} \\ H = H + 360 \quad if H < 0 \end{cases}$$

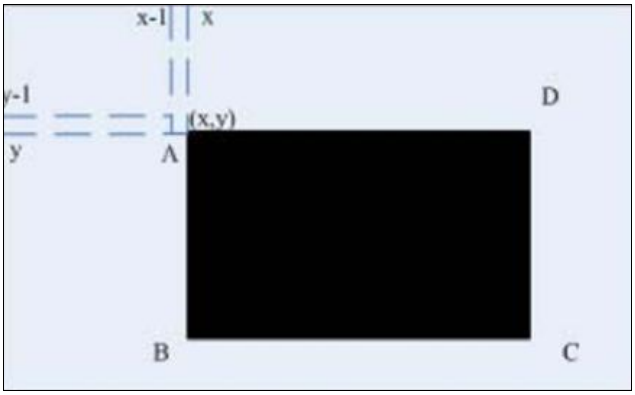
RGB indicates with Red, Green and Blue and HSV denotes HUE, Saturation and intensity.

Step 13-Generating Integral Image

The next step is generation of integral image in KNN as shown in Figure 16. This is done by sum of values immediately and effectively in the rectangular subset of the grid.

$$sat(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y')$$

The main values of any place with (x, y) is the summed area table in the preceding calculation and is a sum of (x, y) from the left. Where i(x^1, y^1) indicates the gray scale value at the point of (x^1, y^1)



Integral Image

Fig 16: Integral Image

With the fact that the value calculated in region table(x, y) the primary summed-area table can be calculated efficiently in a one-pass over the picture.

$$sat(x, y) = sat(x - 1, y) + s(x, y)$$
$$s(x, y) = s(x, y - 1) + i(x, y)$$

S(x, y) shows the combined amount of all pixels in column y above the stage (x, y). S(x,0) = 0 ; Sat(0,y) = 0.

Step 14-Recognizing the number plate by search against the existing database

After identification the number plate text, the text goes for index search in the database as the search crieteria with the number (recognized) as the key. Based on the selection from the database Figure 17 is generated which shows the detected vehicle along with all the details of the vehicle.

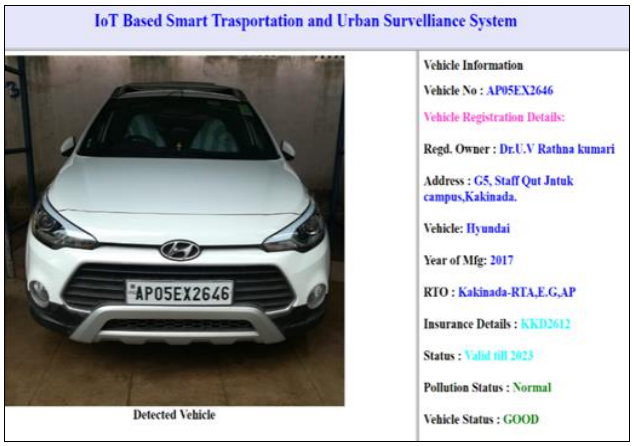


Fig 17: Detecting Vehicle and display of details

If the vehicle details are not found against the database search this vehicle/license plate number is entered as a new entry in the database for future references.

3. Conclusion

There are some iterative demanded situations in which a system model can be able to identify number can be very much useful. This work presents few of those situations, the system designed to match the valuable requirements which are obtained from this approach with also some experimental results which are obtained.

4. References

1. Al Faqheri W, Mashohor S. A Real-Time Malaysian Automatic License Plate Recognition (M-ALPR) using Hybrid Fuzzy. IJCSNS Int J Comput Sci Netw Secur. 2009 Feb;9(2):21-26.
2. Rastegar S, Ghaderi R, Ardeshipr G, Asadi N. An intelligent control system using an efficient License Plate Location and Recognition Approach. Int J Image Process. 2009;3(5):252-258.
3. Ganapathy V, Lui WLD. A Malaysian Vehicle License Plate Localization and Recognition System. J Syst Cybern Informatics. 2008;6(1):1-6.
4. Henriksen JJ. 3D surface tracking and approximation using Gabor filters. South Denmark University, 2007 Mar 28.
5. Martinsky O. Algorithmic and Mathematical Principles of Automatic Number Plate Recognition System. BSc Thesis, Brno University of Technology, 2007.
6. Gonzalez RC, Woods RE. Digital Image Processing. Pearson Education Asia, 2002.
7. Saha S, Basu S, Nasipuri M, Basu DK. License Plate Localization from Vehicle Images: An Edge-Based Multi-stage Approach. Int J Recent Trends Eng. 2009 May;1(1):40-45.
8. Saha S, Basu S, Nasipuri M, Basu DK. An Offline Technique for Localization of License Plates for Indian Commercial Vehicles.
9. Fukunaga K. Introduction to Statistical Pattern Recognition. San Diego: Academic Press, 1990.
10. Gonzalez R, Woods R. Digital Image Processing. Upper Saddle River, NJ: Prentice Hall, 2002.