Research Article

A Novel Traffic System for Detecting Lane-Based Rule Violation

Md. Azmol Fuad[1], Faed Ahmed Arnob[1], Abu Tahir Nizam[1] and Md. Motaharul Islam[2,*]

[1]BRAC University, Dhaka, Bangladesh
azmol14@gmail.com; faed.arnob60@gmail.com; ziantahir@gmail.com
[2]United International University, Dhaka, Bangladesh
motaharul@cse.uiu.ac.bd
*Correspondence: motaharul@cse.uiu.ac.bd

Received: 1st May 2020; Accepted: 9th June 2020; Published: 1st July 2020

Abstract: In recent years, there have been a rise in the number of problems in the existing traffic management system particularly in the developing countries such as Bangladesh, India and Vietnam. Due to this, many accidents have occurred every now and then. Violating the traffic rules such as unpermitted change of lanes and over speeding are the two main reasons for increased number of accidents. In this paper, an attempt is made to solve the problem using Raspberry-pi and OpenCV contour detection technology. A prototype device has been developed to solve the problems regarding lane-based rule violations. The device will be installed in traffic surveillance cameras which will be connected to the metropolitan traffic servers. The device will communicate with the server via Gigabit Ethernet. It will also increase the time efficiency and reduce the manual monitoring cost. Moreover, it will help the traffic management department to find the person responsible for traffic rule violation and assist them to apply the laws strictly. The main contribution of this article is to develop a device that will detect any kind of unpermitted lane changes by any vehicles and identify the license plates. The proposed model has about 80% accuracy according to our evaluation.

Keywords: ***OpenCV contour; License Plate Recognition; Hough Line Transform; KNN Algorithm; Canny Edge detection; Gaussian Blur; Network Attached Storage***

1. Introduction

Nowadays due to excessive road accidents, the governments of different developing countries for example, Bangladesh are paying more attention to solve the lane-based rule violation problem [1]. Currently, the traffic monitoring system of Bangladesh is semi-automatic. The Police and the Traffic monitoring department are trying their heart and soul to perform their task properly but still many violators [2] get away by the loophole of the existing system.

Regardless of the measures taken by the government road transport authority, the number of accidents is still rising. In Bangladesh, the number of road accidents and fatalities has been on the rise compared to those of the years 2014-16. According to the police, 2,635 people had been killed in 2,609 road crashes in 2018. At least 2,513 people had been died in 2,562 accidents in 2017 [1-2]. In addition, the National Advisory group to Secure Transportation, Streets and Railroads (NCPSRR) in Bangladesh had uncovered the data in a press release that 1,552 people had been killed and 3,039 others were injured in 1,495 road accidents across the country in the first four months of 2019 [1-3]. Therefore, a system is needed to detect the rule violator in a more efficient way.

Our proposed model is aiming to take control of this situation with the help of a Raspberry pi-based traffic monitoring system. We are emphasizing in this area as more deaths are occurring due

Md. Azmol Fuad, Faed Ahmed Arnob, Abu Tahir Nizam and Md. Motaharul Islam, "A Novel Traffic System for Detecting Lane-Based Rule Violation", *Annals of Emerging Technologies in Computing (AETiC)*, Print ISSN: 2516-0281, Online ISSN: 2516-029X, pp. 29-41, Vol. 4, No. 3, 1st July 2020, Published by International Association of Educators and Researchers (IAER), DOI: 10.33166/AETiC.2020.03.004, Available: http://aetic.theiaer.org/archive/v4/v4n3/p4.html

to the lane-based rule violation in Bangladesh. Therefore, this needs to be minimized by finding out the culprits and penalizing them accordingly. This will further create awareness among other drivers to follow the rules and maintain them all times. As this is an organized and cost-efficient system, it can be stated as a novel model.

The concept of the whole system is to minimize the number of accidents that occur around us every day and to maintain the traffic rules [2]. In order to do that, a traffic monitoring system is proposed where several algorithms are used. For example- KNN algorithm [4] for neighbouring characters, Gaussian Elimination for Image Blur and smoothness, Hough Line Transform for straight line detection, Canny Algorithm [5-6] to detect a wide range of edges in images, OpenCV Contour [7] for image and video processing. Python is used as the main programming language in OpenCV platform [8].

The major contributions of this article are summarized as follows:

- Implementation of our device on existing traffic cameras to eliminate the cost of setting up new cameras.
- Immediately finding out the culprit who has broken the traffic rule and penalizing the person.
- Use of OpenCV and NumPy as a platform to develop Hough Line Transformation and the License Plate Recognition.
- Our model has around 80% accuracy on an average to complete the License Plate detection and finding out violators.
- Two algorithms have been proposed where one is for detecting road lane and another is for the license plates.
- Use of Raspberry pi 4 for better performance in comparison to Raspberry pi 3 B+ [9].

The rest of the paper is structured as follows. Section 2 discusses the previous research on License Plate monitoring and using it to manage the traffic. The proposed architecture is presented in section 3 and the process of the workflow is also described there. Section 4 describes the algorithms of the proposed system and how those are needed in our proposed model. In section 5, implementation details are stated. Section 6 shows the performance evaluation of the work. Section 7 discusses the future work plan of the proposed system. Finally, Section 8 concludes the paper.

2. Related Works

In [10], authors state that they have discovered a surveillance machine that can be used for management of avenue traffics which is based on Intelligent Visual Internet of Things (IVIoT). With this system via the use of visible tags like license plate numbers, cars' color and automobile types, automobiles can be effortlessly tracked. In addition, it includes other special elements like passing spot and passing moment and also photo haze elimination system. Moreover, they also state that, IVIoT can learn the object in haze weather condition by using a fast and efficient image haze removal method. Besides, some other important features of IVIoT is intelligent visual records mining of visual tags of people or vehicles. These objects can be combined and extended to many fields like public-security-oriented area for the functions of the monitoring of a criminal suspect or a stolen vehicle. In this paper, they show that via the usage of visual sensor nodes which are linked through Wi-Fi network, can identify automobile using visible tags and send the list of blacklisted vehicles to law enforcement authorities. It needs high-resolution cameras, embedded processors, GPS, network adapter and so on. According to the authors, their proposed system has an accuracy of 85.80% on average of real-time tracking.

In article [11], authors work on a project to determine the number of vehicles occupying the streets. They propose a novel system established on fusion of near-infrared imaging signals and demonstrates its capability with theoretical and experimental arguments. They also propose a fuzzy neural network classifier to operate and calculate on the images collected by the near infrared to fulfil the counting function. However, they are using dual-band camera to project their images to make sure they use the range of 0.7 to 2.4 μm in the electromagnetic spectrum. The authors again provide

experimental results for vehicle occupant counter to establish their work to be a robust solution to the problem of automatic vehicle occupant counting.

In article [12], the authors propose an algorithm. Their algorithm includes two major contributions – one is shadow removal method based on Bernsen algorithm combined with the Gaussian filter and another one is character recognition algorithm known as support vector machine (SVM) integration. Their article also provides efficient techniques of image tilt correction and image grey enhancement. According to them, their algorithm is efficient in variance of illumination, view angle, position, size, and color of the license plates when working in a difficult environment. Their algorithm's overall performance of success for the license plate achieves 93.54% when the system is used for LPR in various complex conditions.

In article [13], authors propose a system on how to locate a license plate in any vehicle and shows how important it is to the modern automated transport system. According to them, mostly the license plates have sharp edges and rich texture. Firstly, they extract the information of the vertical edges by image enhancement and sobel operator. Secondly, they remove most of the noise by long curve and random noise removing algorithm. The algorithm is developed by them to find out the plate location and segments for further License Plate reading mechanism. To sum up, the authors of this paper provide experimental results to demonstrate the great robustness and efficiency of their method.

In article [14], authors propose an efficient license plate location and recognition approach through image processing in their research paper. The algorithm includes morphological operators for candidate region. Features of each region will differentiate the license plate from other regions. They also propose color filter for localization, optical character recognition (OCR) for character dilation, multi-layer perception for character recognition and linear vector quantization for comparison. According to their research, they claimed that their proposed system is better in case of low-quality images or images with illumination effects and noise. In article [15], authors propose a new approach for ALPR using sift algorithm for detection and it describes local features of digital images and their own algorithm for character recognition. They claimed that their algorithm has average time of 6.79 seconds for license plate recognition and execution results obtained from their approach gave 88.33% success rates.

However, all of the articles are very much informative as they gave us various approaches for LPR but their works are not related to automatic traffic monitoring and penalizing system. Our work is different from theirs because we are not stopping at license plate recognition (LPR). We are also identifying the accused person who is registered under that license plate violating the traffic rules. Moreover, the metropolitan can penalize and control the traffic rules automatically.

3. Proposed System with Architecture

The proposed system is built on a single board computer which is Raspberry pi. This Pi based prototype model is implemented as shown in Figure 1. Required hardware components for this project are raspberry pi 4, cooling fan, micro SD card and other peripherals. This concludes our proposed device that is attached to the CCTV camera on a traffic signal post.

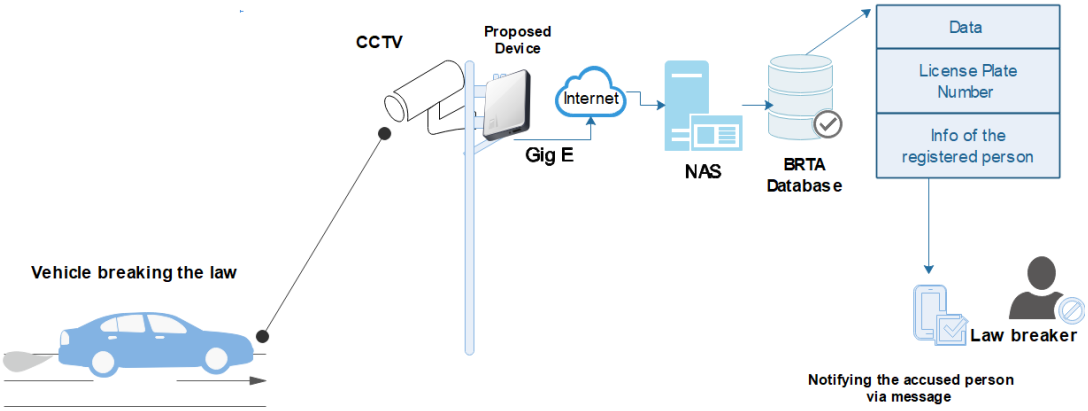

Figure 1. Architecture of Traffic Monitoring System

Then, all the dependencies for OpenCV library of programming functions were installed to work with computer vision. Then python and pip3 were updated for package management system to install and manage software packages written in python version 3. Using pip3, suitable OpenCV version [5] from python has been installed. Some extra dependencies for OpenCV and camera [16-18] were installed too. After that, a flow chart has been developed to regulate and maintain the work process. The following flowchart of Figure 2 shows the working process.

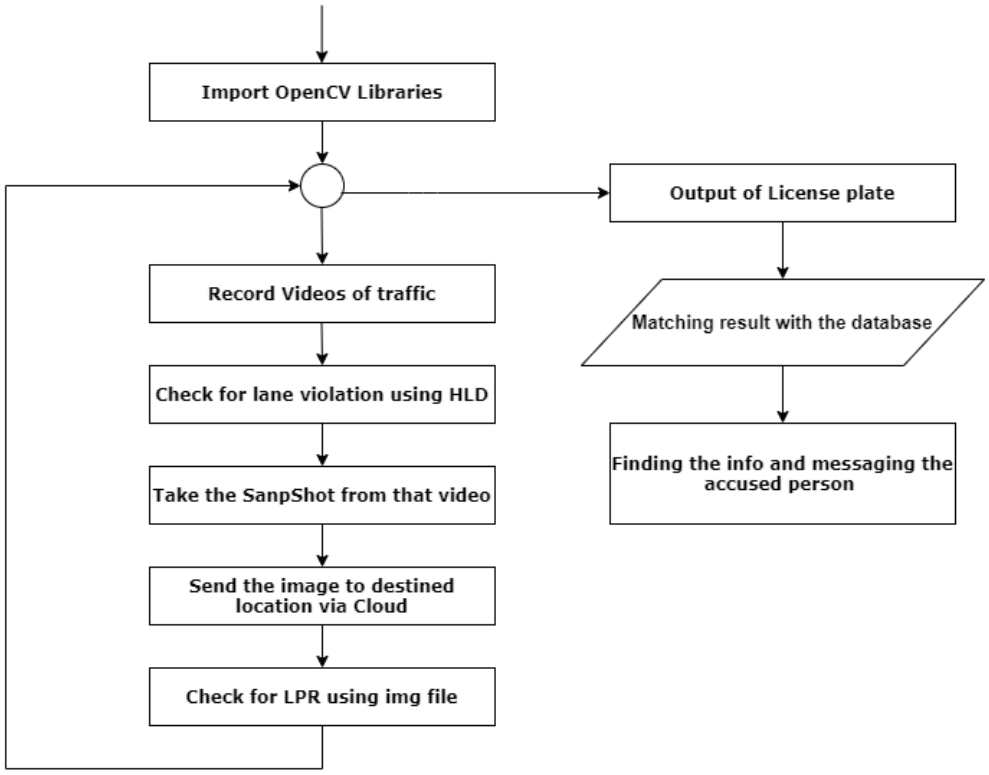

Figure 2. Flowchart of the proposed Traffic Monitoring System

While recording traffic videos according to Figure 2, the main focus is to detect traffic lane violation from the video. In order to do that, some algorithms were followed such as – Hough line transform, gaussian blur and canny edge detection etc. So, first the gaussian blur filter is applied in video. After filtering the video, canny edge detection is done to detect the edge of road lane. Finally, Hough line transform detects the lanes of the road. When this program cannot read any straight line from the videos as a vehicle crosses over the line, it takes a snapshot of that moment and send it to the server. Here algorithm 1 is followed to detect the road lanes and send images to the server.

The server that has been used in the proposed architecture for storage purpose is actually a Network Attached Storage (NAS) server system. As it is a dedicated file storage that enables multiple users and devices to retrieve data. It is an effective system for the proposed model. The captured image is sent to the NAS by the pi from the recordings. The pi sends the data via its gigabit ethernet (Gig E) port. The data sending protocol that is used here is Transmission Control Protocol (TCP). TCP ensures reliable data transmission. This protocol is more suitable here as it is very important that the data should not get lost throughout the process. After the data is sent to the NAS server, the Licence Plate Recognition (LPR) starts.

LPR starts with a series of processes. The first process starts by gray-scaling the scene or image and applying threshold to the image. Image threshold is a simple, yet effective way of partitioning an image into a foreground and background. This image analysis technique is a type of image segmentation that isolates objects by converting grayscale images into binary images. After getting the image with lesser redundant information, OpenCV contour is applied to detect shapes and objects found in that image. A contour is a closed curve of points or line segments, representing the boundaries of an object in an image. In other words, contours represent the shapes of objects found in an image. When a license plate shape is detected, the algorithm takes that shape for further

processing and removes other unnecessary edges and shapes. Next process is to find the vectors of possible characters in the image by the help of NumPy.

Algorithm 1 Hough Line Detection

```
Input: Video(v)
Output: Detected Road Lane(l)
procedure VIDEO CAPTURE(v)
    while true do
        ret, src ←v.read()
        if ret = none then
            procedure VIDEO CAPTURE(v)
                continue
                frame ←GaussianBlur(src, ksize, sigmaX[sigmaY[]])
                hsv ← cvtColor(frame, bgr2hsv)
                lowyellow ←numpy.array([martix low value])
                upyellow ←numpy.array([matrix high value])
                mask ←inRange(hsv, lowyellow, upyellow)
                edge ←Canny(mask, horizontal value, vertical value)
                lines ←HoughLines(edge, hough rho res, hough theta res, hough votes
thresh)
                if lines = true then
                    for lin lines do
                        x1, y1, x2, y2 ←l[0]
                        procedure LINE(frame, (x1, y1), (x2, y2), (0, 255, 0), 5)
                            showframe ←frame
                            showedges ←edge
                            if lines = false then
                                while false do
                                    c = 1
                                    cam = device()
                                    cam.saveSnapshot(c.jpg)
                                    c++
```

Algorithm 1. Algorithm for the line detecting process

NumPy is a general-purpose array-processing package [19]. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with python. Algorithm 1 will store most related characters from the cropped shape to the NumPy array. Finally, the last process will show the most recognized characters in that as output using KNN training in the array. Figure 3 shows the steps in our main.py file.

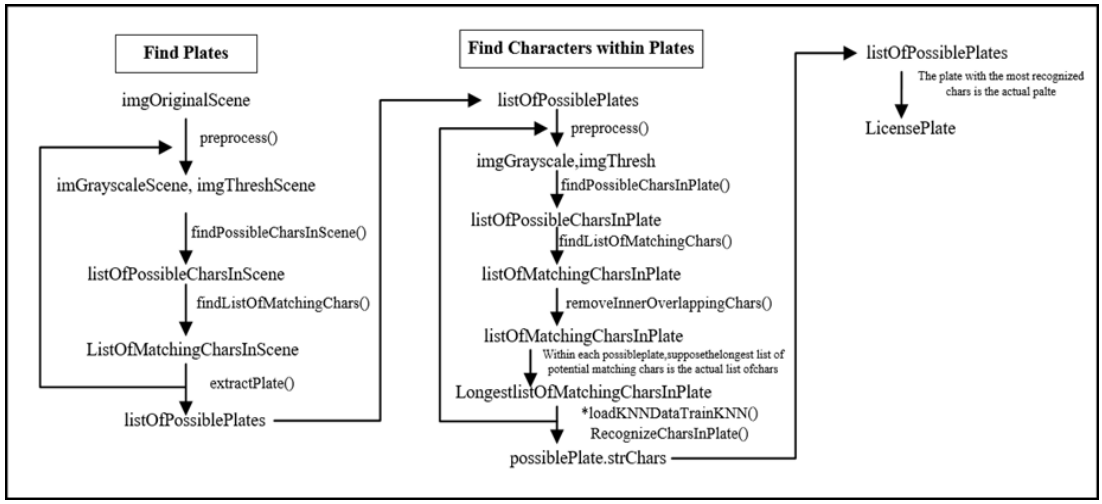

Figure 3. Flowchart of our main.py process

After recognizing the license plate, Bangladesh Road Transport Authority (BRTA) will match the given license plate with their records in the servers and send text message to the owner's contact information about rule violation.

4. Algorithmic Analysis

Hough line transformation is a method to detect any shape. Lines can be represented in two ways – Cartesian Coordinate System and Polar Coordinate System. Hough line transformation uses the polar coordinate system to detect lines from an image. So, representing a line in polar form is –

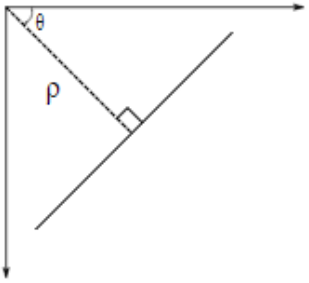

Figure 4. Parametric description of a straight line

$$\boldsymbol{p} = x\cos\theta + y\sin\theta \cdots\cdots\cdots (1)$$

In Figure 4, $\boldsymbol{p}$ is the perpendicular distance from origin in pixels. In Hough line transformation, a 2D array is created to collect evidences of lines in an image. This 2D array is called an accumulator. After that, Canny edge detection [6] is used in the project to check for edges in the collected 2D accumulator array. For every edge pixels of the accumulator it varies the values of $\theta(0-\pi)$ and put it in equation (1) to get the value of $\boldsymbol{p}$ which will show the lines in the image.

If we take 100X100 image for example with a horizontal line at the middle and take the values (x, y) of first point of the line and put the values of θ = 0,1,2…,180 and check the value of $\boldsymbol{p}$, in accumulator, the cell (50,90) = 1 along with some other cells. Now if the same process is done for second point of the line, in accumulator this time, the cell (50,90) = 2. Here, we are voting the ($\boldsymbol{p}$, θ) values. If we search the accumulator for maximum votes, we will get the value (50,90) which represents there is a line in this image at distance 50 from origin and at angle 90 degrees.

Image noises demonstrates itself as random variations in the brightness or color of pixels in an image or speckles similar to analogue camera's film grain. To reduce noises and extra sharpness in the video, blurring techniques of OpenCV has been used. Here, Gaussian blur technique is also used as video obtains various high frequency contents. Other than box-filtering, Gaussian blur uses a Gaussian kernel [20] where width and height have to be mentioned. Gaussian blurring algorithm will filter over each pixel of the picture and recalculate the pixel esteem based on the pixel values that encompass it. The range checked each pixel is called a kernel. A bigger kernel filters a bigger sum of pixels that encompass the centre pixel. In Python, cv2.GaussianBlur() function operates with three parameters-

dst = cv2.GaussianBlur(**src**, **ksize**, **sigmax**[**dst**[,**sigmaY**[borderType=Border Default]]]) ……. (2)

Where dst is the output image and **src** is the input image. The size of the **sigma** of the function dictates how wide the curve should be inside the kernel. **Ksize** represents the matrix that the algorithm uses to scan over the image. This algorithm is used to reduce the noise and the sharpness in terms of our Hough Line detection process.

In Figure 5, the image represents a python logo with black background. When it is observed more precisely, some amount of blue and yellow colors having white color dots at the edge are also seen. However, a computer sees the image in a large 2-dimensional matrix of numbers where each number represents the color of a pixel. If the image noise is expressive enough, it can potentially interfere with computer vision system's functionality. If gaussian blur is applied according to equation (2) with a kernel size of (5, 5), we will see the result in Figure 6.

To continue with Hough line detection, edge of lines needs to be detected. For that Canny edge algorithm [5-6] [8] has been used. It has many stages to it. First step is noise reduction by

GaussianBlur(). After that, the edge gradient and direction of each pixel is done by filtering and using Sobel kernel [5]. Next step is to remove the unnecessary pixels that contain the edges based on magnitude, direction and angle of the gradient. For that, every pixel needs to be checked for maximum edges in its neighbourhood in terms of the direction of the gradient. Finally, all the edges are checked with hysteresis threshold value to find if they are real edges. In thresholding two values are set - max and min. Any gradient of the edge having more than max value are considered real edges. Others that are below min value are ignored. If any edges lie between the two values, they are chosen as classified edges or non-edge in terms of their connectivity.

Figure 5. Original Image

Figure 6. Blurred image

The K-nearest neighbours (KNN) algorithm is a basic yet supervised machine learning algorithm. It is used to solve both classification and regression problems. It is also used to recognize characters in the license plate. KNN works by finding the distances between a query and all the examples are there in the data. Then the algorithm selects the specific examples (K) closed to the query and votes for the most prominent label.

To sum up, the mentioned processes have been followed to develop algorithm 2 as the main algorithm which takes input of image generated from algorithm 1 and shows the output of the license plate from the image-

Algorithm 2 Main Process

```
Input: Image(img)
Output: Llicense Plate(lp)
procedure KNN TRAINING
    if KnnTaining = Flase then
        print error
    procedure OPEN IMAGE(img)
        if img = None then
            print error
    procedure DETECT PLATES(img)
        procedure DETECT CHARACTERS(detect plates)
            if detect plates = 0 then
                "No Plates Found"
            else
                Sort Lists of Possible Plates In Descending Order
                x = sorted lists[0]
                procedure CROP OF PLATE(x)
                    procedure THRESHOLD(x)
                if detect characters = 0 then
                    "No Characters Detected"
                procedure DRAW RECTANGLE AROUND PLATE(img,x)
                procedure WRITE LICENSE PLATE ON IMAGE(img,x)
                    showsceneimage ←img
                    lp ←write(img)
```

Algorithm 2. Algorithm for the main process

5. Implementation Details

The proposed device was implemented and tested based on collection of test data. This process of the testing consisted of many software and hardware components. A working architecture of the prototype is provided in Figure 9.

The specifications of the main device Raspberry pi 4 is given in Table 1. Initial setup was completed by connecting a camera to the Raspberry pi. Pi was booted up with Raspbian OS via Secured Shell. OpenCV [8] and all the dependencies of it have been installed to work with the lane violation. For testing the proposed model, we have provided a test video of a vehicle breaking the lane-based rule, as shown in Figure 7.

Algorithm 1 has been developed for detecting lane-based rule violation. The test video has been imported to find out the lanes and if the vehicle has violated the rule or not by the algorithm 1. The violation will be detected if the Hough lines are unable to construct a perfect straight line. In Figure 7, the straight line of the solid lane has been blocked by a vehicle. During the test, a motorcycle has been used. The lane detection process has been disrupted as shown in left side of Figure 7. The program has not been able to create a perfect line which was there before. As soon as the violation is detected the algorithm captures that frame as shown in right side of Figure 7. Then the image is sent to the NAS server. The data is sent via TCP protocol.

Figure 7. Lane Violation detection (Left) and Frame Capture (Right)

In the next step, image processing takes place in order to recognise the license plate. For that, algorithm 2 has been implemented. It starts with grey-scaling the image and applying threshold to the image. Then, OpenCV contour has been applied to detect shapes and object found in that image. When a license plate shape is detected, the algorithm takes that shape.

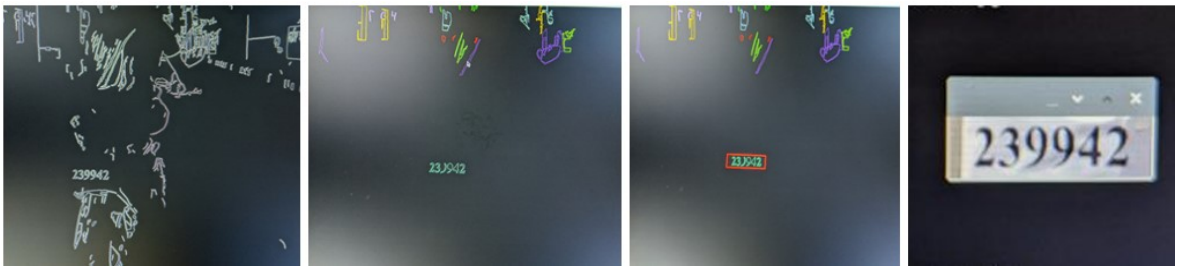

Figure 8. Plate detection process

Table 1. Tools and Components' Specification

Components	Specification
System on Chip (SoC)	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC
CPU	1.5 GHz 64-bit quad-core ARM
RAM	4GB LPDDR4-3200 SDRAM
Wi-Fi	Dual-band 802.11ac (2.4 GHz and 5GHz)
Ethernet	Gigabit Ethernet over USB 2.0 (max 300 Mbps). Power-over-Ethernet support (with separate PoE HAT)
General Purpose Input/ Output (GPIO)	40-pin (fully backwards compatible with previous boards)
Power	5V/2.5A DC power input
Operating System	Linux / Unix
Video Processor	H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)

To detect characters from that selected shape, NumPy is used to find out the vectors of possible characters [4] in every contour that was found. In the multidimensional array, every possible character is stored. In that array, KNN training is applied to find out the most related characters of the license plate. After finishing the process, it gives the license plate number as output shown in Figure 8.

For finding out the vehicle owner, the plate number is checked with the BRTA database to get the contact information. Finally, as BRTA holds the owner information, the authority will be notified with the detected license plate and the authority will send message to the owner about rule violation, as shown in Figure 9.

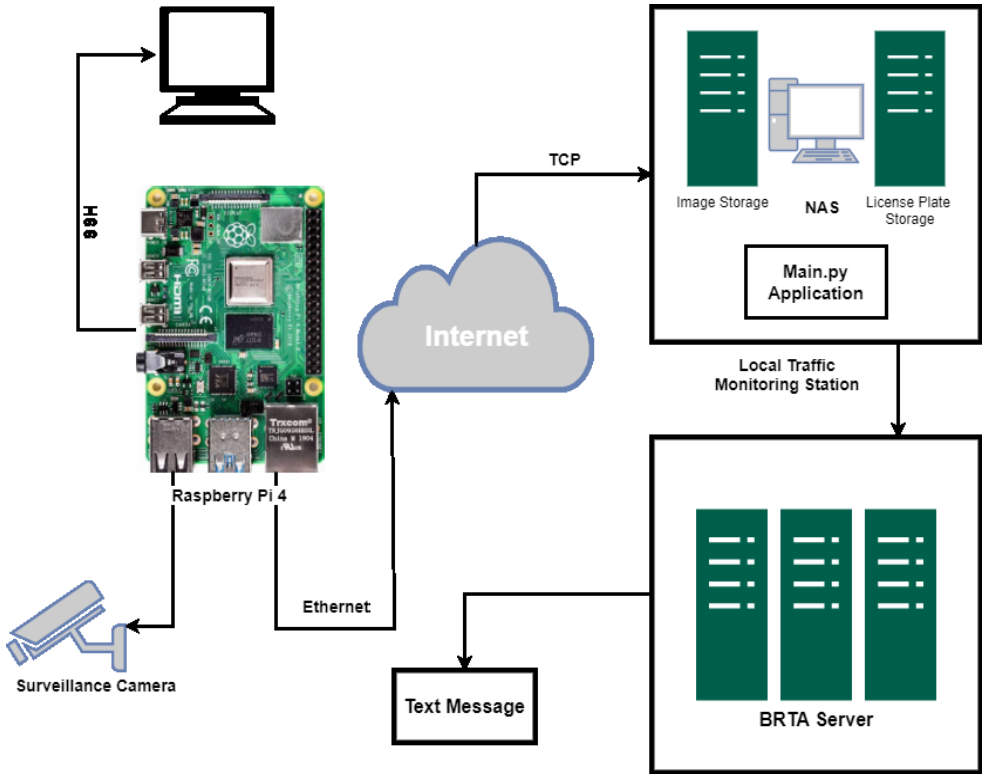

Figure 9. Architecture of the Implemented System

6. Performance Evaluations

This section discusses the efficiency of the proposed system. License plate detection accuracy and time consumption for plate detection are the main parameters for the evaluation. In terms of plate recognition, the accuracy is needed because false result may refer to different license plate. As the proposed system deals with moving vehicles the timing is very important here. The lane detection and the plate recognition should be in the minimum amount of time.

Figure 10. Images that are used in this accuracy test

To check the performance of the model, some images have been provided in Figure 10. The graph in Figure 11 shows the accuracy of our developed License Plate Recognition System. The Y-axis shows the percentage of the accuracy and the X-axis shows the different types of images. The system has faced a major drawback in image 3 where the system could not recognize the license plate. Rather it has recognized the brand logo of Audi as a valid license plate number.

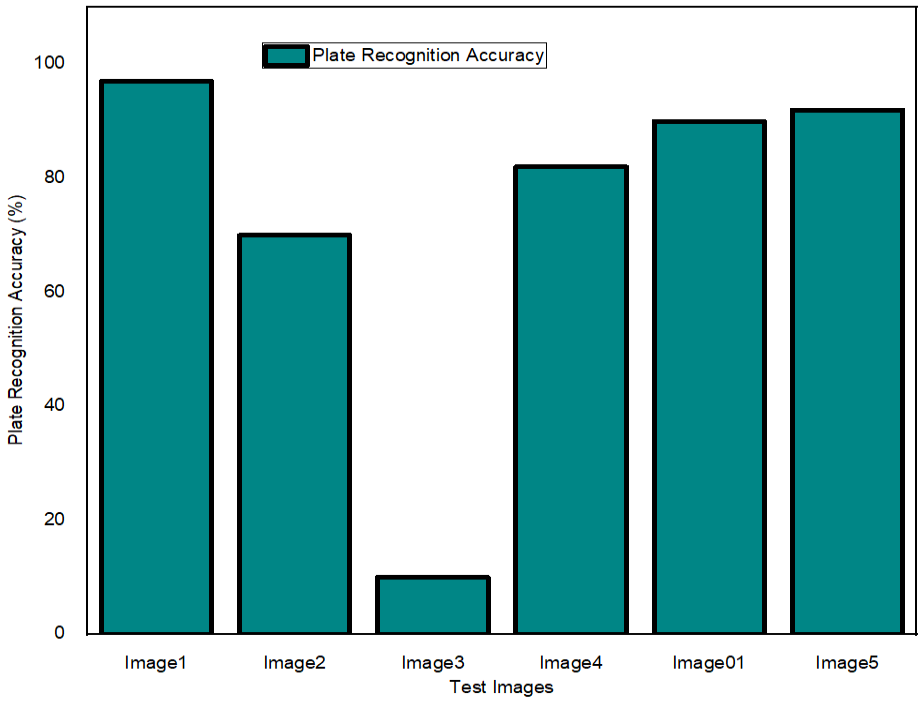

Figure 11. Showing the accuracy percentage of our model

Besides, the model also has faced problems with the angle of the plate. Whereas many established algorithms on LPR has a higher success rate of more than 92% [21]. Our system has about 80% accuracy in most of the cases.

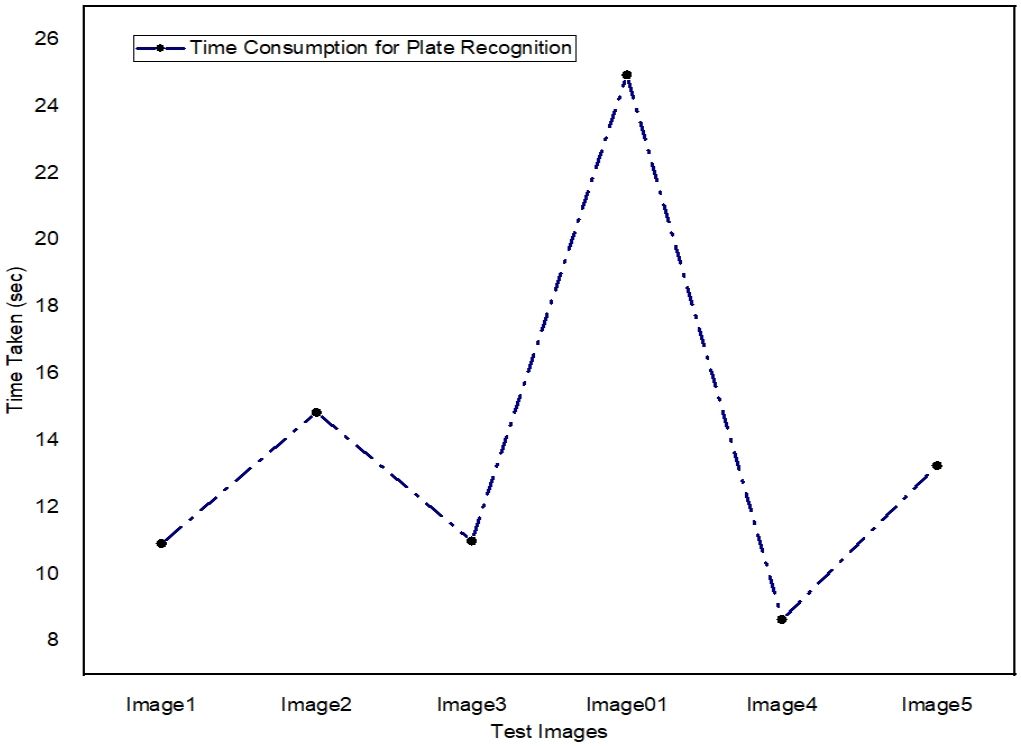

Figure 12. Evaluating the time consumption of the recognition process

Timing is very important in terms of detecting the plate correctly. The lesser the time needed to detect plates, the better the algorithm is. Here, Figure 12 shows the time consumption of LPR for test images.

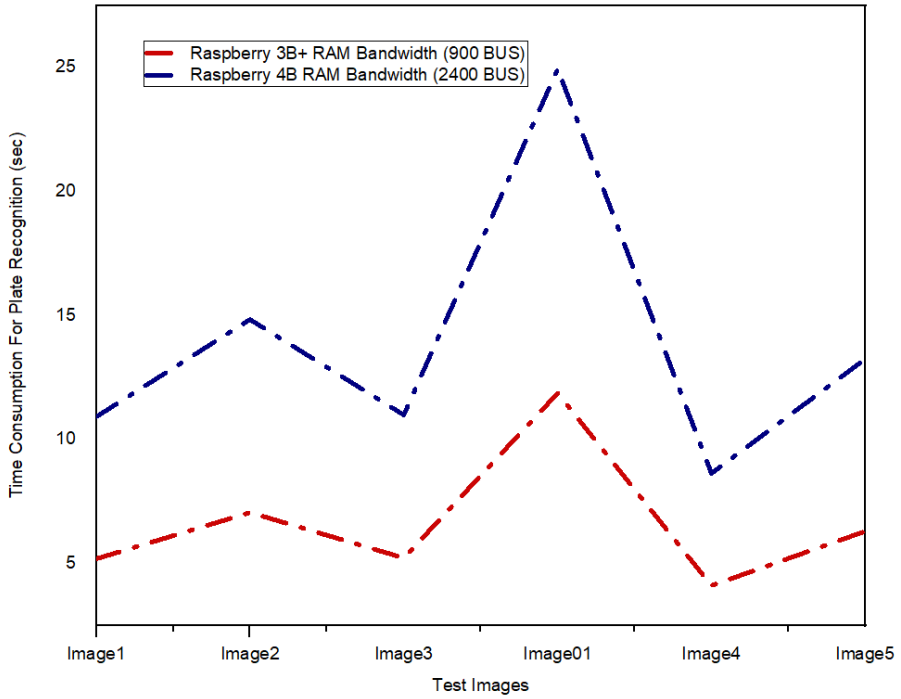

Figure 13. Change in performance of the time consumption of the system in terms of RAM Bandwidth.

During this evaluation, sample pictures have been used from Figure 10. These pictures are of different sizes and of different pixel densities. The higher the file size of the picture, the clearer the image is. In the Figure 12, the image01 has the highest file size of them all and the plate is a little bit tilted away from the camera. For this reason, the image 01 is taking around 24 seconds to get the license plate detected. On the other hand, image 4 has the lowest file size, lowest pixel density and the plate is just in front of the camera position resulting only 8 seconds to get detected.

On our previous work [9] we developed our system based on Raspberry Pi 3 B+ which was a 1 gigabyte RAM variant. Although having a minimum amount of RAM the whole system has been able to perform task efficiently but having more memory space can do the work better and faster.

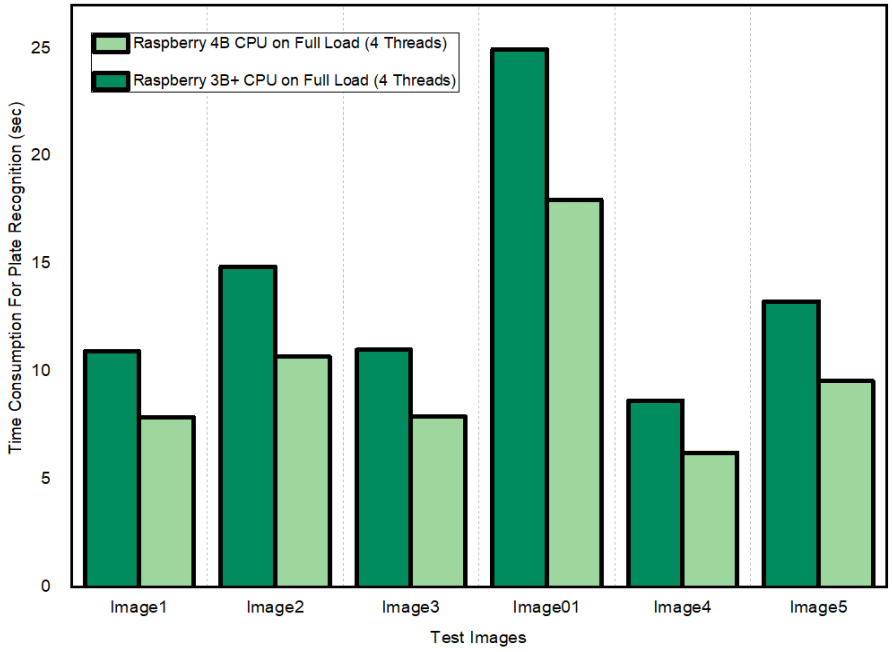

Figure 14. Change in performance of the time consumption of the system in terms of CPU load.

As this is an extended work of ours [9], Pi 4 has been used in this performance evaluation test. In Figure 13, we can see the performance improvement is about 2.1 times faster while using Raspberry pi 4. This will be more beneficial in large-scale traffic monitoring. On the other hand, Figure 14 represents the CPU computation time which is about 27% faster than the previous model.

7. Future Works

This system aims to be a perfect system for which some future works is needed. In the performance evaluation section, it is seen that the proposed system has about 80% accuracy in terms of serving a proper result. However, to achieve the maximum rate of accuracy, Deep Neural Network can be implemented in future. There are also some latency issues in terms of detection process and sending file to the server. Here, EDGE computing can be applied to solve that problem. Besides, there are also weather issues such as hazy, rainy weather conditions etc and human ethics error such as covering the license plate etc. So, there is a very good scope of improvement in detecting license plate before real life deployment. Enforcement of all the mentioned work can lead this model to a great potential.

8. Conclusions

A well- functioning city needs law-abiding citizens. The law enforcement authority should be able to apply the law upon the citizens in order to maintain a city. Our proposed system can assist the law enforcement authority and the traffic-monitoring department in this context. The accuracy level of the proposed system is sufficient to find the accused vehicles' license plates and to recognize the responsible person according to the registration plate number. Thus, law enforcers can bring them under the judicial system. Moreover, in terms of Dhaka city, we can reduce the number of accidents by penalizing the lawbreakers. By penalizing one accused we can alert a thousand more so that they do not break the rules and cause accidents. This is our vision that in near future our proposed system will help the Traffic Management Department to make the streets safer for the citizens.

Acknowledgements:

We would like to pay our gratitude to the anonymous reviewers for their very useful comments that helped us enrich the quality and presentation of the paper a lot.

References

[1] "NCPSRR: 1,212 killed in road accidents in three months" *Dhaka Tribune*, 02 April 2019. Available: https://www.dhakatribune.com/bangladesh/nation/2019/04/02/ncpsrr-1-212-killed-in-road-accidents-in-three-months.

[2] T. S. Adhikary, "Road crashes on rise despite govt measures" *The Daily Star*, 20 March 2019. Available: https://www.thedailystar.net/frontpage/news/road-crashes-rise-despite-govt-measures-1717696.

[3] "Road Accidents: '1,552 killed, 3,039 hurt in four months'" *The Daily Star*, 12 May 2019. Available: https://www.thedailystar.net/frontpage/news/road-accidents-1552-killed-3039-hurt-four-months-1742266.

[4] O. Harrison, "Machine Learning Basics with the K-Nearest Neighbors Algorithm" *Medium*, 14 July 2019. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.

[5] S. Sahir, "Canny Edge Detection Step by Step in Python - Computer Vision" *Medium*, 27 January 2019. Available: https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123.

[6] Ali Akbar Shah, Bhawani S. Chowdhry, Tayab D. Memon, Imtiaz H. Kalwar and J. Andrew Ware, "Real Time Identification of Railway Track Surface Faults using Canny Edge Detector and 2D Discrete Wavelet Transform", Annals of Emerging Technologies in Computing (AETiC), Print ISSN: 2516-0281, pp. 53-60, Vol. 4, No. 2, 2020. Available: http://aetic.theiaer.org/archive/v4/v4n2/p5.html.

[7] Madhav, "*OpenCV Image Segmentation: Tutorial for Extracting specific Areas of an image*", CircuitDigest, 13 March 2019. Available: https://circuitdigest.com/tutorial/image-segmentation-using-opencv.

[8] A. Mordvintsev and A. K, "*OpenCV-Python Tutorials Documentation Release 1*", 20 March, 2014. Available: http://www.nitc.ac.in/electrical/ipg/python/opencv-python-tutroals.pdf.
[9] Faed Ahmed Arnob, Md. Azmol Fuad, Abu Tahir Nizam, Shuvajit Barua, Ahnaf Atef Choudhury and Md. Motaharul Islam, "An Intelligent Traffic System for Detecting Lane Based Rule Violation" International Conference on Advances in the Emerging Computing Technologies 2019, Al Jamiah, Medina 42351, Saudi Arabia, 10-12 February, 2020.
[10] Q. Li, H. Cheng, Y. Zhou, and G. Huo, "Road vehicle monitoring system based on intelligent visual internet of things," Journal of Sensors, vol. 2015, 6 July 2015.
[11] I. Pavlidis, V. Morellas, and N. Papanikolopoulos, "A vehicle occupant counting system based on near-infrared phenomenology and fuzzy neural classification" IEEE Transactions on intelligent transportation systems, vol. 1, no. 2, pp. 72–85, 2000.
[12] Y. Wen, Y. Lu, J. Yan, Z. Zhou, K. M. von Deneen, and P. Shi, "An algorithm for license plate recognition applied to intelligent transportation system" IEEE Transactions on Intelligent Transportation Systems, vol. 12, no. 3, pp. 830–845, 2011.
[13] D. Zheng, Y. Zhao, and J. Wang, "An efficient method of license plate location" Pattern recognition letters, vol. 26, no. 15, pp. 2431–2438, 2005.
[14] S. Rastegar, R. Ghaderi, G. Ardeshipr, and N. Asadi, "An intelligent control system using an efficient license plate location and recognition approach" International Journal of Image Processing (IJIP), vol. 3, no. 5, pp. 252–264, 2009.
[15] F. A. da Silva, A. O. Artero, M. S. V. de Paiva, and R. L. Barbosa, "Alprs-a new approach for license plate recognition using the sift algorithm", arXiv, 7 March 2013. Available: https://arxiv.org/abs/1303.1667.
[16] Md. Motaharul Islam, Khan, Zaheer and Alsaawy, Yazed, "A framework for Harmonizing Internet of Things (IoT) in Cloud: Analyses and Implementation", Wireless Networks, Springer, 2019.
[17] Sun-Min Hwang, Kyu-Jin Kim, Md. Motaharul Islam et. al. "Multi-Modal Sensing Smart Spaces Embedded with WSN Based Image Camera", In the 3rd International Conference on Pervasive Technologies Related to Assistive Environments, June 23 - 25, 2010, Samos, Greece.
[18] Md. Motaharul Islam, Eui-Nam Huh, "Sensor Proxy Mobile IPv6 (SPMIPv6)-A Novel Scheme for Mobility Supported IP-WSNs", Sensors, vol. 11, no. 2, pp. 1865-1887, 2011.
[19] E. Bressert, "SciPy and NumPy: An Overview for Developers", O'Reilly Media, Inc., 2012.
[20] K. J. Millman and M. Aivazis, "Python for scientists and engineers", Computing in Science & Engineering, vol. 13, no. 2, pp. 9–12, 2011.
[21] B. Moharil, V. Ghadge, C. Gokhale, and P. Tambvekar, "An efficient approach for automatic number plate recognition system using quick response codes" IJCSIT, vol. 3, no. 0975-9646, pp. 5108–5115, 2012.