978-1-7281-4452-8/20/$31.00 ©2020 IEEE

An Intelligent Traffic System for Detecting Lane Based Rule Violation

Faed Ahmed Arnob
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
faed.arnob60@gmail.com

Md. Azmol Fuad
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
azmol14@gmail.com

Abu Tahir Nizam
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
ziantahir@gmail.com

Shuvajit Barua
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
shuvajitbarua6@gmail.com

Ahnaf Atef Choudhury
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
ahnaf.atef@bracu.ac.bd

Md. Motaharul Islam
Computer Science and Engineering
United International University
Dhaka, Bangladesh
motaharul@cse.uiu.ac.bd

***Abstract*- In recent years, there have been rise in the number of problems in the existing traffic management system particularly in the developing countries. Due to this, many agonizing accidents are occurring every now and then. Over speeding and violating the traffic rules such as unnecessary change of lanes are the two main reasons for the rise in the number of accidents. This problem needs to be solved immediately to reduce the number of unexpected deaths. In this paper, an attempt is made to solve the problem addressed using Raspberry-pi and OpenCV contour detection technology. We have developed a prototype device to solve this problem. The device will be installed in traffic surveillance camera near the traffic signal position, which will be connected to the metropolitan traffic servers. If any of the vehicle crosses the device violating the mentioned traffic rule, they will be detected and the data will be sent to the server immediately. Thus, the proposed traffic monitoring system will help to reduce the manual collection of data, resulting less time wastage and this will further reduce the cost. Furthermore, it will help to find the person responsible for traffic rule violation and will assist the traffic management department to apply the laws strictly. The proposed model has about 78.83% accuracy, which will help to reduce the number of accidents that are taking place every day.**

Keywords— OpenCV contour, Traffic Management, LPR, Hough Line Transform, KNN Algorithm, Canny Edge detection, Gaussian Blur

I. INTRODUCTION

Nowadays due to excessive road accidents, the government of different developing countries is paying highest level of attention to solve the problems [1] in the existing traffic management systems. The current situation of the traffic monitoring system is semi-automatic. Police and Traffic department are monitoring the situation. They are trying their heart and soul to do their task properly but still many violator [2] gets away by the loophole of the existing process. Therefore, an automatic process to detect people breaking the traffic rules with minimum error, less time wastage and least possible cost is necessary.

Our proposed model is aiming to take control of this massacre with the help of an intelligent traffic monitoring system. It is able to detect any kind of lane-based traffic rule violation, which is the first and foremost reason of any accidents. We are focusing more in this area because the number of deaths in the streets is rising in an alarming rate. Therefore, this needs to be minimized quickly by finding out the culprits and penalizing them accordingly. This will further alert other drivers to follow the rules and maintain them at all times.

The concept of the whole article is to minimize the number of accidents that happens around us every day and also to maintain the traffic rules. In order to do that, [3] we have implemented an Automatic License Plate Recognition (ALPR) System. Here we have included several algorithms – KNN Algorithm [14] for neighboring characters, Gaussian Elimination for Image Blur and smoothness, Hough Line Transform for straight line detection, Canny Algorithm [13] to detect a wide range of edges in images, OpenCV Contour [9] for image and video processing, Tesseract Library for character [12] recognition, and finally Python3 [16] was used as a center of all algorithms.

Regardless of the measures taken by the administration following a phenomenal understudy development for street security, the number of street accidents and fatalities had expanded a year ago compared with those of the earlier year. According to police, 2,635 people were killed in 2,609 road crashes last year. At least 2,513 people had died in 2,562 accidents in 2017 [1] [2].

In addition, the National Advisory group to Secure Transportation, Streets and Railroads (NCPSRR) uncovered the data in its ordinary month to month overview and perception report that 1,212 individuals, including 157 ladies and 215 youngsters, were killed and 2,429 others harmed in street mishaps in the country over the initial three months of this current year. Also, a sum of 1,168 lethal mishaps occurred on different expressways and local streets between January 1 and 31, 2019 [1] [2].

In order to prevent this massacre, our contributions based on our proposed idea are:

- We have implemented our device on existing traffic cameras to minimize the cost of setting up a new camera.
- Immediately finding out the culprit who has broken the traffic rule and penalize the person.
- We have used OpenCV and NumPy as a platform to develop the Hough Line Transformation and the License Plate Recognition.
- Our Model has around 78.83% accuracy on average to complete the License Plate detection and finding out the violator.

978-1-7281-4452-8/20/$31.00 ©2020 IEEE

The rest of the paper is structured as follows. Section II discusses the previous research on License Plate monitoring and using it to manage the traffic and how our model is different from them. The whole architecture is presented in section III and the process of our workflow is also described there. Section IV describes the algorithms of our proposed system and how those are needed in our proposed model. In section V we have stated the Implementation details. Section VI shows the performance evaluation of our work. Eventually, section VII concludes our paper.

II. Related Works

In [4] 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 [5] 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 gray 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 and 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 [6] 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 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 provides experimental results to demonstrate the great robustness and efficiency of their method.

In [7] author 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 that will differentiate the license plate from other regions, 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 [3] author state that they discover 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 number, cars color and automobile type, automobiles can be effortlessly tracked. In addition, it includes special other elements like passing spot and passing moment and also photo haze elimination system. Moreover, they also state that, IVIoT is a great intelligent visible tag because it can learn the object from a tremendous distance. 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 officers. It needs high-resolution cameras, embedded processors, GPS, network adapter and so on.

In [8] 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 is very fast in real time and results obtained from their approach gave very good success rates.

However, all of the articles are very much informative as they gave us various approach for LPR but their works are not related to automatic traffic controlling 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.

III. Proposed System With Achitecture

Fig.1 represents the architecture of the proposed system. The following subsections describe the whole process of our proposed architecture.

A. Components

The Raspberry Pi based traffic monitoring system of our proposed system is built on a single chip computer that works on LINUX operating system with a camera. Required hardware components for this project: Raspberry Pi 3 Model-B, Cooling Fan, Cam and other peripherals, Micro SD Card. This concludes our proposed device that is attached to the surveillance camera on a traffic signal post.

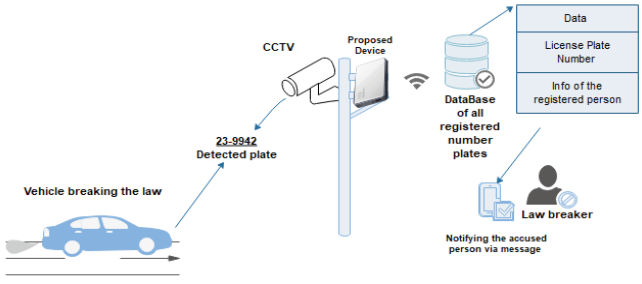

Fig.1. Architecture of Traffic Monitoring System

B. Proccedure of our Proposed system

B.1- Booting up Raspberry Pi: We have sketched an architecture for building the project at first. To make our raspberry pi up and running, we have installed Raspbian Buster OS in 32 GB micro SD card which will be used as storage of our raspberry pi. While the micro SD boots up, we added a 60 mm cooling fan over the BCM2387 chipset processor powered up from GPIO 40 pin. After Raspbian OS boots up, we update and upgrade the OS with necessary software updates.

B.2- Installing OpenCV and OpenCV Contour: After that, we installed all dependencies for OpenCV library of programming functions of computer vision. Then we updated our python and pip3 for package management system to install and manage software packages written in python version 3. Now using pip3, we have installed suitable [13] OpenCV from python. We also installed extra dependencies for OpenCV and Camera. After that, we have developed a flowchart that shows the prototype of our project that we will make. The following flowchart of Fig.2 shows the proposed system.

B.3- LPR from Image File: We have detected license plates from images by our python main source code using OpenCV. The process starts by gray-scaling the scene or image and then apply threshold to the image.

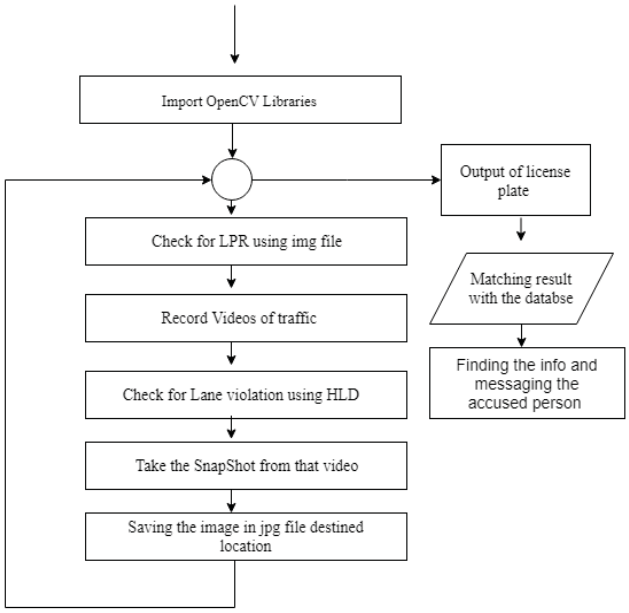

Fig.2 Flowchart of the proposed Traffic Monitoring System

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 that, we tried to find all the possible character in the scene or image and contour all of them. 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. Then we tried to find the vectors of possible characters in the image by the help of NumPy.

NumPy is a general-purpose array-processing package. 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. It finds the shape of all possible license plates and crops them off from the original image and again it runs the same process starting from the Image Grayscale and Image Thresh. Finally, our main.py file shows the plate with the most recognized chars in that as output. Fig.3, show the steps in our main.py file.

B.4- Record videos of traffic via surveillance camera: For recording videos, we developed an algorithm where we set the standard video dimensions and also the type of videos such as .avi and .mp4 etc. After that, to grab the resolution of our camera we check the capability of our cam whether it can match with the standard video dimensions. If they match, our surveillance cam will turn on and start recording frame by frame. To have a copy of the recorded video, we have made a function in our python code where it will save the recorded video.

B.5- Check for Lane Violation by using HLT, Gaussian Blur and Canny Edge: Here we have focused on traffic lane violation from recorded videos. In order to do that, we have followed some algorithm such as – Hough line Transform, Gaussian blur and Canny edge detection.

So, first we have applied the Gaussian blur filter in our captured video. After filtering the video, we have masked the image from where we can import the image and export it to Canny edge detection. Then we have applied the Hough line transform to detect the lines of the road. While 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 LPR section to scan the vehicle's license plate. Then we have matched the license plate number [11], which we got from our model with the license plate database to find the person's information that is registered under that plate number.

B.6- Send Text Message: To send text message to the person who violated the rules, we have developed an algorithm where we have used a message server and a database to match the license plate number and details to send a message to the person's contact number.

IV. Algorithms Of Proposed System

A. Hough Line Transformation

Hough line Transformation is a method to detect any shape. As we know, we can represent line 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 -

$$p = x\cos\theta + y\sin\theta \quad (1)$$

Here p is the perpendicular distance from origin in pixels. In Hough Line Transformation, a 2D array known as accumulator is created to collect evidences of lines in an image. After that we used Canny Edge Detection for our project to check for edges in the collected 2D accumulator array. Then, for every edge pixels of the accumulator we

vary the values of $\theta(0-\pi)$ and put it in equation (1) to get the value of p which will show the lines in the image.

B. Gaussian Blur

Images and videos can contain different types of noise and extra sharpness because of camera sensor. To reduce noises and extra sharpness in our video. We used one of the blurring techniques of OpenCV. We used Gaussian blur technique as our recorded video obtained various high frequency contents. Other than box-filtering, Gaussian blur uses a Gaussian kernel [16] where width and height have to be mentioned. If not mentioned, it will take kernel size from the ksize parameter. In Python, cv2.GaussianBlur() function operates with several parameters-

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

Where, dst is the output image and src is the input image. The deviation of X and Y direction (sigmaX, sigmaY) will be taken from K size if it is not mentioned in the function. We have used this algorithm to reduce the noise and the sharpness in terms of our Hough Line detection process.

D. Text message

We have hired a trial host and developed this python file shown in Fig.4 in order to send accused person who is breaking the traffic rule.

```
from twilio.rest import Client
account_sid =
'ACb8db5eda469c18adfa5ad2bd1979d423'#
auth_token =
'341ab29333f69814042aa8a425410f5f'#
myPhone = '+8801521200380' # Phone
number you used to verify my account
TwilioNumber = '+12055399996' # Phone
number given to you by Twilio
client = Client (account_sid,
auth_token)
client.messages.create(
to=myPhone,
from_=TwilioNumber,
body='Dear_____, you have been
penalized for breaking the traffic
rules. Your vehicle no is 2399422.'+
u'\U0001f68
```

Fig.4 Code snippet of the text server we have developed

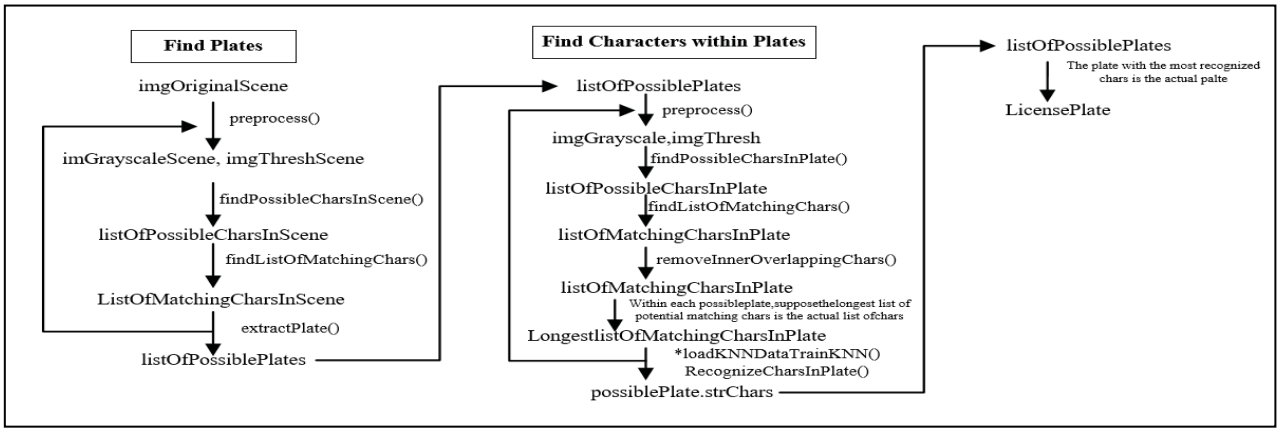

Fig.3 Flowchart of our main.py process

C. Canny Edge Detection

To continue with Hough Line detection, edge of lines needs to be detected. For that we used Canny Edge algorithm [13] [12]. It has many stages to it. First step is noise reduction by GaussianBlur(). After that, we need to get the edge gradient and direction of each pixels which is done by filtering and using Sobel Kernel [13]. Based on magnitude, direction and angle of the gradient next step is to remove the unnecessary pixels that contain the edge. For that every pixel needs to be checked for maximum edges in its neighborhood in terms of the direction of the gradient. Finally, all the edges are check with hysteresis thresholding to find if they are really edges are not. 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.

E. Algorithm for the main process

Input: Image from the captured video

Output: Best possible match of character in license plate

1. Image pre-processed via Gray-Scaling and Threshold. Methods: preprocess()
2. Finding possible characters in the image. Method: findPossibleCharsInScene()
3. Listing matching characters in the image. Method: findListsOfListsOfMatchingChars()
4. Extracting list of possible plates. Method: extractPlate()
5. Repeat method 1,2,3 in list of possible plates
6. Remove overlapping characters in List of matching characters in plate

7. Apply KNN algorithm to detect nearest and longest potential match with the charters in plate. Method: recognizeCharsInPlate()
8. Extract most recognized license plate from the list of possible plate

V. Implementation Detail

We have implemented and tested our developed prototype based on demo data. This process of our testing consisted of many software and hardware. In Fig.5 we can see our prototype's working architecture. We have installed Raspberry Pi 3 as the System on a Chip (SoC) of our project and connected the necessary peripherals. As the System on a Chip (SoC) heats up pretty quickly we added a 60mm cooling fan for our proposed device. Connecting a camera and input output devices such as keyboard, monitor, mouse completed our setup for the task. We booted up our prototype device on Raspbian OS and installed OpenCV [12] and all the dependencies of it to work with the video output that we will get from our camera. We have also recorded a demo video of a vehicle breaking the lane-based rule.

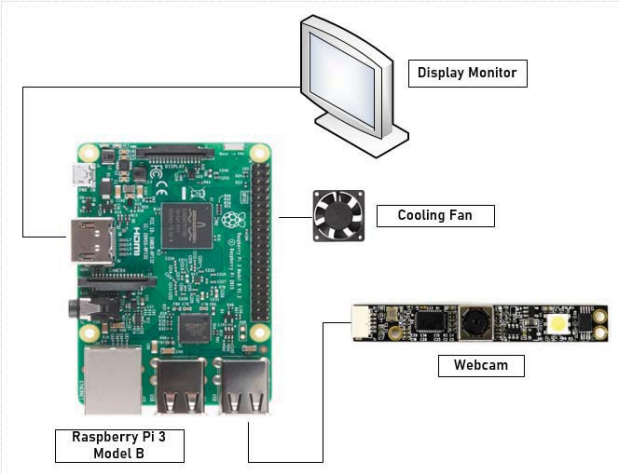

Fig.5 *Architecture of the Implemented System*

For our programming section we have also installed movie.py dependencies to work with the video output that we will get from our camera. We have recorded a demo video of a vehicle violating the lane-based rule with a visible license plate. Then we imported that video to our Hough Line Transform (HLT) section to find out the lanes and if the vehicle has violated the rule or not. The violation will be detected if the Hough Lines are unable to construct a perfect straight line. As soon as the violation gets detected our system captures that frame and saves it to our destined location. Next Step is image processing in order to recognise the license plate. It starts with gray-scaling the image and applying threshold to the image. This is an highly effective process to get rid of redundant background or foregorund. After getting image with lesser redundant information we applied OpenCV contour to detect shapes and obeject found in that image. We have used NumPy to find out the vectors of possible charecters [14] in every contours that we have found. NumPy [15] provides a high-performance multidimentional array processing package to work with these values that we get from the image and finally Fig.3 process completes and gives the result of our desired license plate. After finishing the process mentioned above, it gives us the license plate number and for finding out the vehicle owner we match the palte number with our created dummy datatbase to get the contact information. Finally via our developed text message server we notified the person via a text message.

VI. Performance Evaluation with Graphical Comparison

The graph Fig.6 shows the accuracy of our developed License Plate Recognition System. These images are test samples Fig.7, which were tested by our program. The Y-axis shows the percentage of the accuracy and the X-axis shows the different types of images. We faced a major drawback of our system in image 3 where our system didn't recognize the license plate rather than it recognized the brand logo of Audi as a valid license plate number. Other

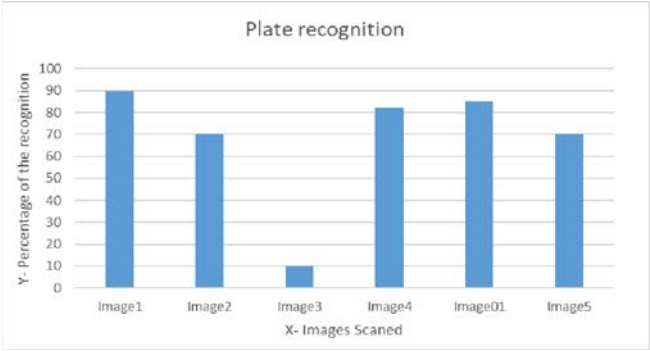

Fig.6 Showing the accuracy percentage of our model

then that we face some problem here and there with the angle of the plate. Whereas many established algorithms which worked on LPR only has a higher success rate of more than 92% every time (discussed in the related works section). Our system has about 78.83% accuracy in most of the cases. However, which we are not just providing LPR system in our project. We have developed a complete traffic monitoring system, which makes us different from other related works. Here Fig.7 are the images that we have used consecutively

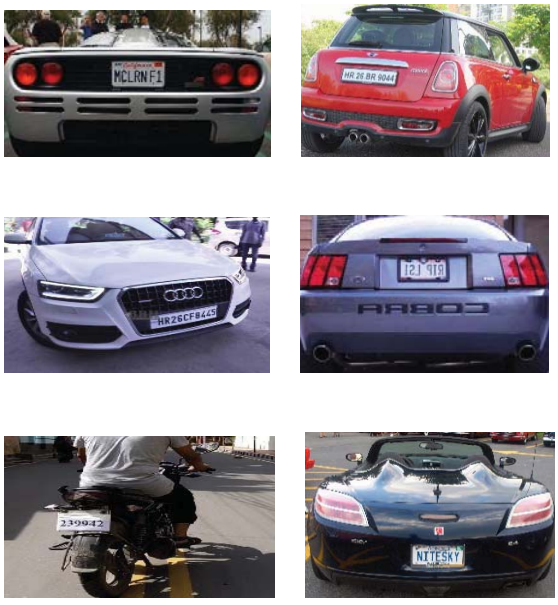

Fig.7. Images that are used in this accuracy test

Timing is very important in terms of detecting the plate correctly. Generally, the lesser the needed time to detect plates the better the algorithm is. Surely there are [10] some real-time detection systems, which are used on a super computer, but our project is based on Raspberry Pi. Therefore, it does take some time to recognize the plates. However, from this evaluation we have found the ways of how to make it more time efficient during the evaluation of Fig.6.

Fig.8 Evaluating the time consumption of our recognition process

During this evaluation of ours we have used the same sample pictures that were used in Fig.6 evaluation. These pictures are of different sizes and of different pixel densities. The higher the file size of the picture the clearer the image is. Resulting the pixel density to be higher also. In the Fig.7 the image 01 has the highest file size of them all and the plate is a little bit far 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 has the plates just in front of the camera position. Resulting only 8 seconds to get detected.

VII. Conclusions

A perfectly functioning city needs law-abiding citizens and the law enforcers should be able to apply and enforce the law upon all the citizens in order to make a city well functioning. Our developed prototype can help the law enforcers and the traffic-monitoring department to some extent. Although, 78.83% is not perfect but it is more than enough to find the accused vehicles license plates and find the accused person according to the registration plate number. Thus, we can bring them under the justice. Moreover, in terms of our 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 one day our device will help the Traffic Management Department to make the streets safer for us.

References

[1] "NCPSRR: 1,212 killed in road accidents in three months," *Dhaka Tribune*, 02-Apr-2019. [Online]. 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-Mar-2019. [Online]. Available: https://www.thedailystar.net/frontpage/news/road-crashes-rise-despite-govt-measures-1717696.

[3] 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, 2015.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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 preprint arXiv:1303.1667, 2013.

[9] M., *OpenCV Image Segmentation: Tutorial for Extracting specific Areas of an image*. [Online]. Available: https://circuitdigest.com/tutorial/image-segmentation-using-opencv.

[10] 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.

[11] A. Raj, *Car License Plate Recognition using Raspberry Pi and OpenCV*. [Online]. Available: https://circuitdigest.com/microcontroller-projects/license-plate-recognition -using-raspberry-pi-and-opencv.

[12] A. Mordvintsev and A. K , "*OpenCV-Python Tutorials Documentation Release 1*," Available: http://www.nitc.ac.in/electrical/ipg/python/opencv-python-tutroals.pdf. 2019.

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

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

[15] E. Bressert, SciPy and NumPy: an overview for developers. " O'Reilly Media, Inc.", 2012.

[16] K. J. Millman and M. Aivazis, "Python for scientists and engineers," Computing in Science & Engineering, vol. 13, no. 2, pp. 9–12, 2011.

[17] M. O. Hasan, M. M. Islam and Y. Alsaawy, "Smart Parking Model based on Internet of Things (IoT) and TensorFlow," *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, Sarawak, Malaysia, Malaysia, 2019, pp. 1-5.

[18] S. M. Faiaz Mursalin, Meherin Hossain Nushra, Quazi Ashikur Rahman, Nashita Binte Asad, Miah Mohammad Asif Syeed and Md. Motaharul Islam, "Indoor Car Parking with the Assistance of Line following Robot" *International Conference on sustainable technologies for industry 4.0*, Dhaka, Bangladesh, December 24-25, 2019