

AUTOMATIC NUMBER PLATE RECOGNITION (ANPR)

Muhammed Faiz bin Abdul Kader

A170522

A170522@siswa.ukm.edu.my

Abstract: Automatic Number Plate Recognition (ANPR) is an image processing technology which uses number plate to identify the vehicle. The main objective of this project is to design and develop effective image processing techniques and algorithms to localize the number plate in the captured image using Java programming. Some of the approach like noise reduction, grayscale conversion, edge detection, binarization, finding contour and image segmentation is used to crop the number plate location. Finally, to evaluate the performance of system, the localization is tested with a number of vehicle images. An accuracy of more than 80% was achieved for the localization of vehicle number plates.

Keywords: Number plate, number plate recognition, localization, image processing, Java programming

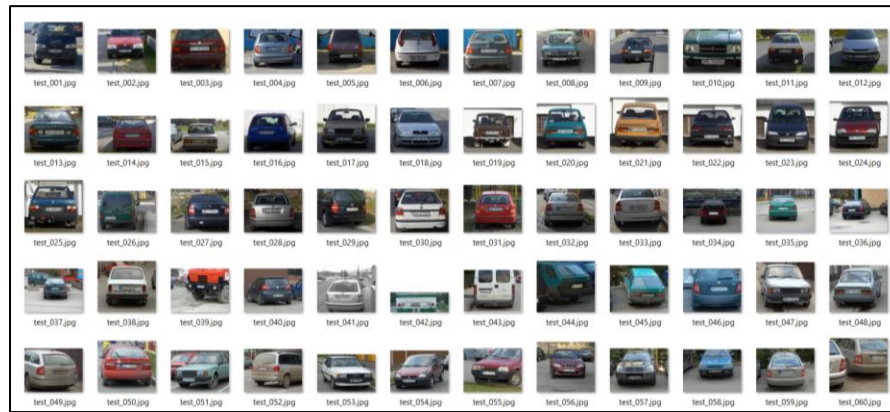
1 INTRODUCTION

Number plate recognition system is a system which aided human in reading vehicle number plate in an efficient way in order to save time and energy. Image processing technology is one of the techniques involved to treat an image as a two-dimensional signal and applying standard signal-processing to it. This technique is a crucial technique in number plate recognition system to read the number plate from a vehicle image. Automatic number plate recognition system has a wide range of applications since the number plate of vehicles is the most primary, widely accepted, human readable, mandatory identifier of vehicles. This system has played a significant rule in traffic law enforcement and it is widely applied over the world to identify stolen cars based on the up-to date blacklist. There are other useful applications such as vehicle access control, automatic toll collection, real-time monitoring and parking area security and management. The most dominant and basic step is to determine the exact location of the number plate in the image. In order to obtain an appropriate recognition, the number plate recognition technique consists of two main methods which are image acquisition and image pre-processing.

2 METHODS

2.1 IMAGE ACQUISITION

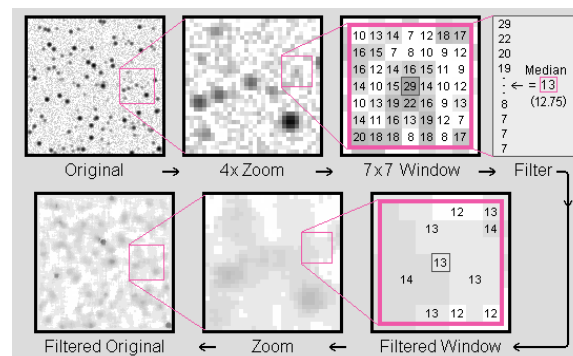
The data collected for this work are vehicle images. Precisely, 97 vehicle images were acquired from an online website which is provided by Ondrej Martinsky. The acquired images are colored Jpeg images and are stored in a folder on the system used for the development. Out of the 97 images, 10 images were used for system testing.



2.2 IMAGE PRE-PROCESSING

2.2.1 Noise Reduction

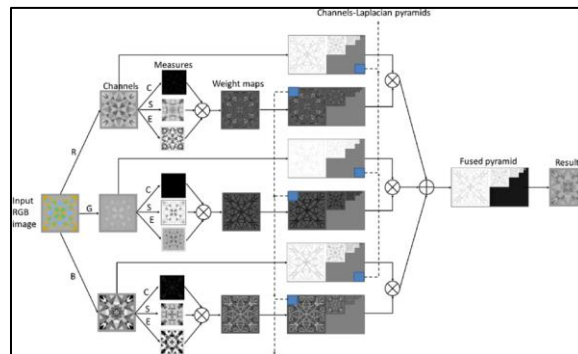
The median filter is the filtering technique used for noise removal from images and signals. Median filter is very crucial in the image processing field as it is well known for the preservation of edges during noise removal. The prior duty of the filter is to scan every input data interceding the overall entries with the median function known as “window” method. The window tends to be a little bit complex over the higher-dimensional signals. The number of medians is set according to the number of windows, falling under odd and even categories.





2.2.2 Grayscale conversion

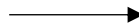
The luminosity method is a more sophisticated version of the average method. It also averages the values, but it forms a weighted average to account for human perception. We're more sensitive to green than other colors, so green is weighted most heavily. The formula for luminosity is $0.21 R + 0.72 G + 0.07 B$.



2.2.3 Edge detection

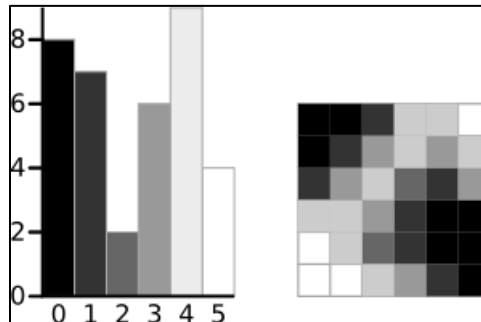
The sobel operator is applied to each point in the image. At each point a horizontal and vertical filter is applied to detect changes in intensity of this point compared to its direct neighbour points. A high difference in intensity of side-by-side point implies then a part of an edge. On the other side, if a point and its neighbour points have the same grayscale value, the operator value will be zero which means no change in intensity and thus no possible edge. The sobel operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the gradients.

$$\begin{array}{c}
 \frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \\
 S_x \quad S_y \\
 \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \\
 \text{Sobel} \quad \text{weighted average and scaling} \quad \text{x-derivative} \\
 \text{Run filter over image} \\
 \frac{\partial f}{\partial x} = S_x \otimes f \quad \frac{\partial f}{\partial y} = S_y \otimes f \\
 \text{Image gradient} \\
 \nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]
 \end{array}$$

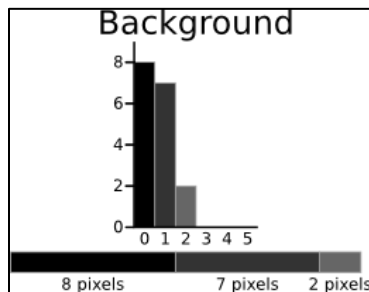


2.2.4 Binarization

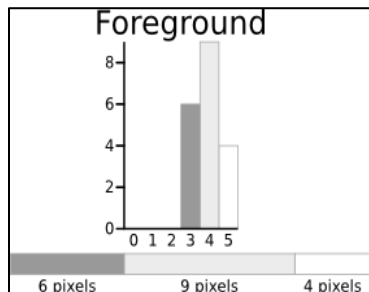
Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum. The algorithm will be demonstrated using the simple 6x6 image shown below. The histogram for the image is shown next to it. To simplify the explanation, only 6 greyscale levels are used.



The calculations for finding the foreground and background variances (the measure of spread) for a single threshold are now shown. In this case the threshold value is 3.



$$\begin{aligned} \text{Weight } W_b &= \frac{8 + 7 + 2}{36} = 0.4722 \\ \text{Mean } \mu_b &= \frac{(0 \times 8) + (1 \times 7) + (2 \times 2)}{17} = 0.6471 \\ \text{Variance } \sigma_b^2 &= \frac{((0 - 0.6471)^2 \times 8) + ((1 - 0.6471)^2 \times 7) + ((2 - 0.6471)^2 \times 2)}{17} \\ &= \frac{(0.4187 \times 8) + (0.1246 \times 7) + (1.8304 \times 2)}{17} \\ &= 0.4637 \end{aligned}$$

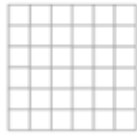
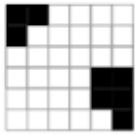
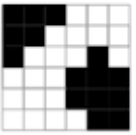
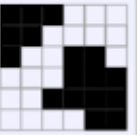
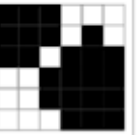
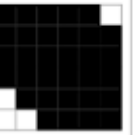
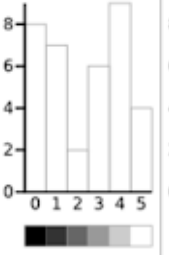
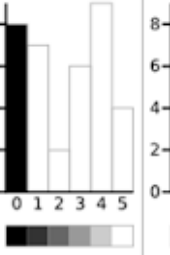
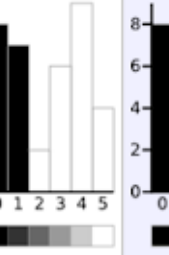
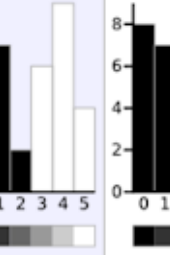
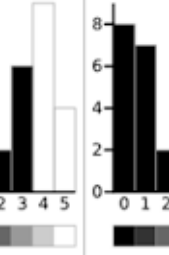
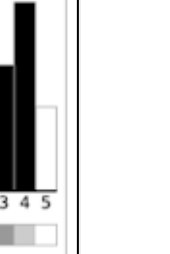


$$\begin{aligned} \text{Weight } W_f &= \frac{6 + 9 + 4}{36} = 0.5278 \\ \text{Mean } \mu_f &= \frac{(3 \times 6) + (4 \times 9) + (5 \times 4)}{19} = 3.8947 \\ \text{Variance } \sigma_f^2 &= \frac{((3 - 3.8947)^2 \times 6) + ((4 - 3.8947)^2 \times 9) + ((5 - 3.8947)^2 \times 4)}{19} \\ &= \frac{(4.8033 \times 6) + (0.0997 \times 9) + (4.8864 \times 4)}{19} \\ &= 0.5152 \end{aligned}$$

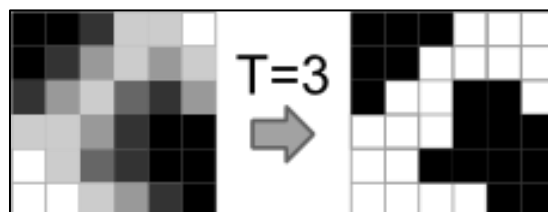
The next step is to calculate the 'Within-Class Variance'. This is simply the sum of the two variances multiplied by their associated weights.

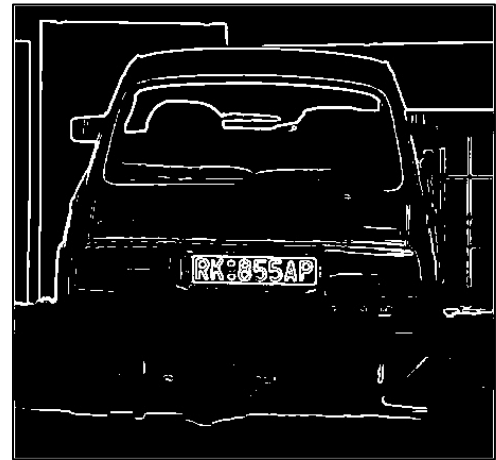
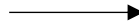
$$\text{Within Class Variance } \sigma_W^2 = W_b \sigma_b^2 + W_f \sigma_f^2 = 0.4722 * 0.4637 + 0.5278 * 0.5152 = 0.4909$$

This final value is the 'sum of weighted variances' for the threshold value 3. This same calculation needs to be performed for all the possible threshold values 0 to 5. The table below shows the results for these calculations. The highlighted column shows the values for the threshold calculated above.

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
						
						
Weight, Background	$W_b = 0$	$W_b = 0.222$	$W_b = 0.4167$	$W_b = 0.4722$	$W_b = 0.6389$	$W_b = 0.8889$
Mean, Background	$M_b = 0$	$M_b = 0$	$M_b = 0.4667$	$M_b = 0.6471$	$M_b = 1.2609$	$M_b = 2.0313$
Variance, Background	$\sigma_b^2 = 0$	$\sigma_b^2 = 0$	$\sigma_b^2 = 0.2489$	$\sigma_b^2 = 0.4637$	$\sigma_b^2 = 1.4102$	$\sigma_b^2 = 2.5303$
Weight, Foreground	$W_f = 1$	$W_f = 0.7778$	$W_f = 0.5833$	$W_f = 0.5278$	$W_f = 0.3611$	$W_f = 0.1111$
Mean, Foreground	$M_f = 2.3611$	$M_f = 3.0357$	$M_f = 3.7143$	$M_f = 3.8947$	$M_f = 4.3077$	$M_f = 5.000$
Variance, Foreground	$\sigma_f^2 = 3.1196$	$\sigma_f^2 = 1.9639$	$\sigma_f^2 = 0.7755$	$\sigma_f^2 = 0.5152$	$\sigma_f^2 = 0.2130$	$\sigma_f^2 = 0$
Within Class Variance	$\sigma_W^2 = 3.1196$	$\sigma_W^2 = 1.5268$	$\sigma_W^2 = 0.5561$	$\sigma_W^2 = 0.4909$	$\sigma_W^2 = 0.9779$	$\sigma_W^2 = 2.2491$

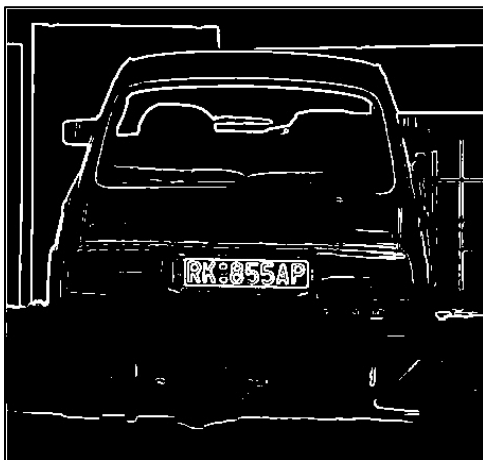
It can be seen that for the threshold equal to 3, as well as being used for the example, also has the lowest sum of weighted variances. Therefore, this is the final selected threshold. All pixels with a level less than 3 are background, all those with a level equal to or greater than 3 are foreground. As the images in the table show, this threshold works well.

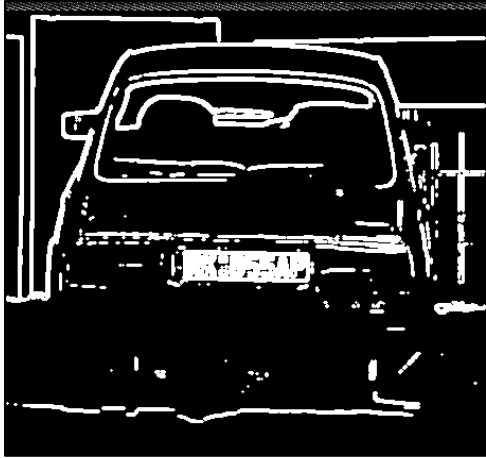


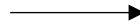


2.2.5 Finding contour

A contour is a closed curve joining all the continuous points having some color or intensity, they represent the shapes of objects found in an image. Contour detection is a useful technique for shape analysis and object detection and recognition. After performing edge detection and binarization, we find the points where the intensity of colors changes significantly and then we simply turn those pixels on. However, contours are abstract collections of points and segments corresponding to the shapes of the objects in the image. As a result, we can manipulate contours in our program such as counting number of contours, using them to categorize the shapes of objects, cropping objects from an image (image segmentation) and much more.

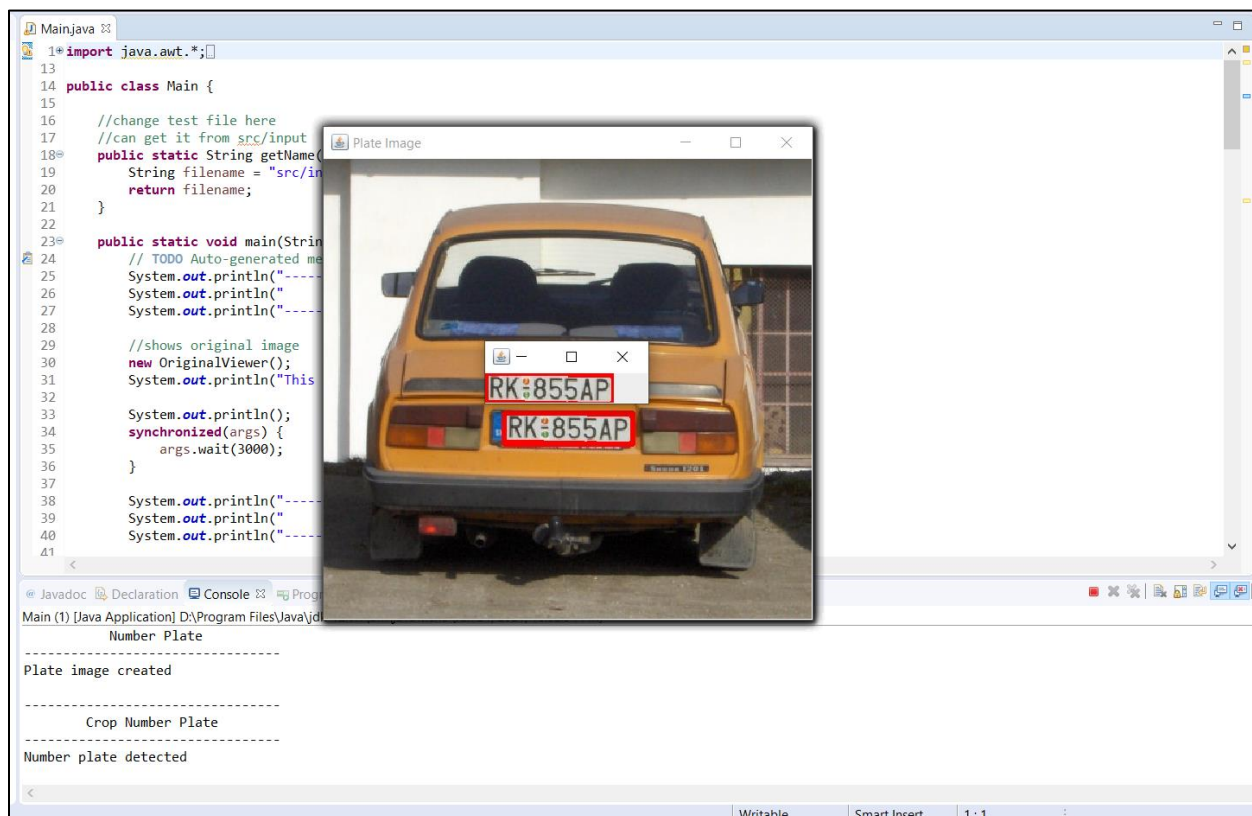


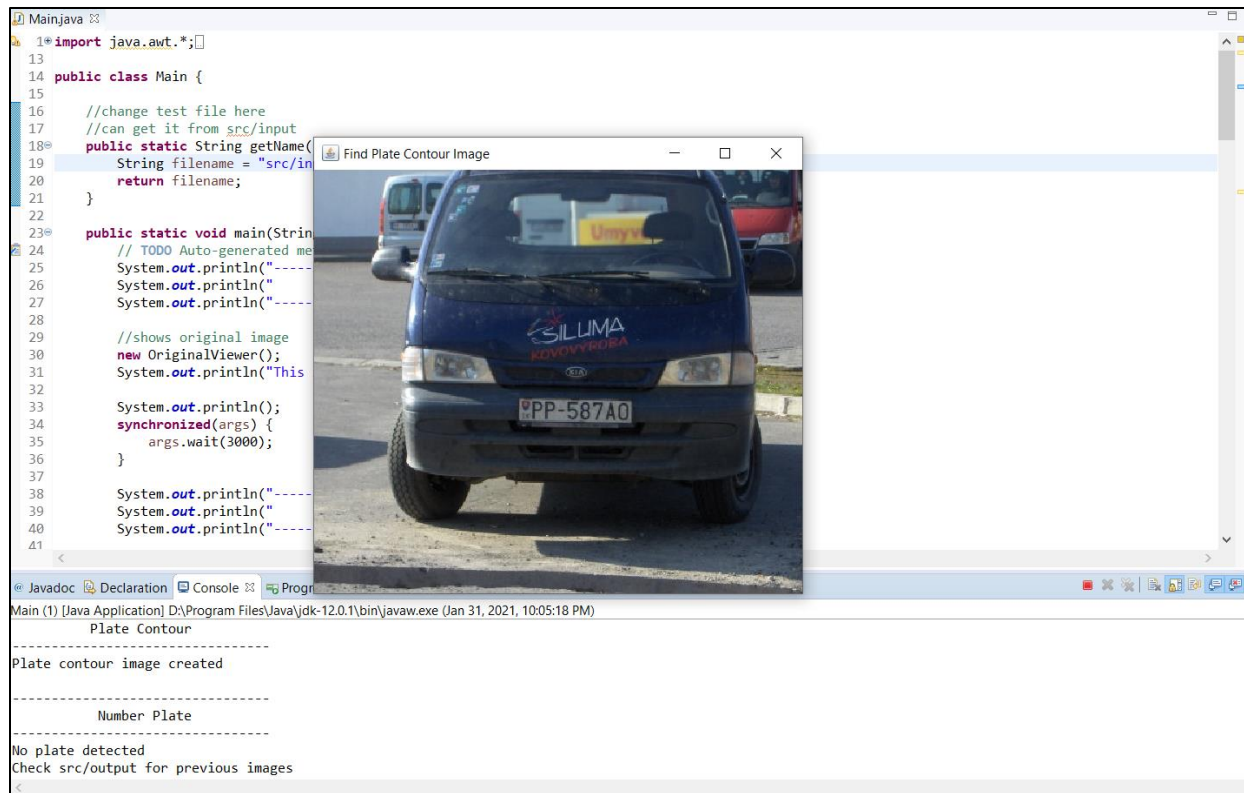




3 FINDINGS AND ARGUMENTS

Among the 10 images that had been tested, 9 of the number plates from the vehicle images were successfully recognize while 1 of them failed to recognize the number plate. This is due to some factors that might give some difficulties for the program to recognize the number plate such as the angle, backgrounds, false edges and others. Below are one of the results for the successful and failure to recognize the number plate.





4 CONCLUSION

In this paper, the automatic number plate recognition system is presented. The system uses a series of image processing techniques for recognizing the number plate of vehicle images. The system is implemented in Java and its performance is tested on real images. The simulation results shows that the system robustly detect and recognize number plate although there are some failures. The implementation works quite well however, there is still room for improvement.

5 REFERENCES

- [1] Sahil Shaikh, Bornika Lahiri, Gopi Bhatt and Nirav Raja, "A novel approach for Automatic Number Plate Recognition". 2013 International Conference on Intelligent Systems and Signal Processing (ISSP).
- [2] M. Tahir Qadri and M. Asif, "Automatic Number Plate Recognition System for Vehicle Identification Using Optical Character Recognition". 2009 International Conference on Education Technology and Computer.

- [3] Tella Pavani and DVR Mohan, "Number Plate Recognition by using open CV- Python". Volume: 06 Issue: 03, 2019, IRJET.
- [4] Esmat Rashedi and Hossein Nezamabadi-pour, "A hierarchical algorithm for vehicle license plate localization". Multimed Tools Appl (2018).
- [5] Jia Wang, Boris Bacic and Wei Qi Yan, "An effective method for plate number recognition". Multimed Tools Appl (2018).
- [6] Leonard G. C. Hamey and Colin Priest, "Automatic Number Plate Recognition for Australian Conditions". Proceedings of the Digital Imaging Computing: Techniques and Applications (DICTA 2005).
- [7] Ondrej Martinsky, "Algorithmic and Mathematical Principles of Automatic Number Plate Recognition Systems". BRNO 2007.
- [8] Hamid Mahini, Shohreh Kasaei, Faezeh Dorri and Fatemeh Dorri, "An Efficient Features-Based License Plate Localization Method". 2006 IEEE.
- [9] K. M. Sajjad, "Automatic License Plate Recognition using Python and OpenCV".
- [10] Anu Agarwal and Sudhir Goswami, "An Efficient Algorithm for Automatic Car Plate Detection & Recognition". 2016 Second International Conference on Computational Intelligence & Communication Technology.
- [11] Amr Badr, Mohamed M. Abdelwahab, Ahmed M. Thabet and Ahmed M. Abdelsadek, "Automatic Number Plate Recognition System". Volume 38(1), 2011.
- [12] Ng Simin and Florence Choong Chiao Mei, "Automatic Car-plate Detection and Recognition System". EURECA 2013.
- [13] Er. Kavneet Kaur and Vijay Kumar Banga, "Number Plate Recognition Using OCR Technique". Volume: 02 Issue: 09, Sep-2013.