Automated Parking Lot Management for Bengali License Plates Using Hough Transformation and Image Segmentation

Abul Ahsan Mahmud

(10201029)

Bishal Peter Dores

(10201021)

Nahid Ul Islam

(10201001)

Supervisor

Rubel Biswas

Co - Supervisor

Md. Zahangir Alam

Department of Computer Science & Engineering

August 2014

BRAC University, Dhaka, Bangladesh

Abstract

This paper is on automated parking lot management system for Bengali language based on vehicle number plate recognition. The need for an efficient parking system stems from increased congestion, motor vehicle pollution. The aim of this research is to develop and implement an automatic parking system that will increase convenience and security of the parking lot with minimum human involvement. This is done with the combination of automated license plate recognition and database manipulation. The image is acquired from a live video feed by comparing frames for significant changes. Then it is passed through several steps including some different types of algorithms and methodologies such as Canny Edge Detection, Connected Component Labeling and Hough Transformation to localize the license plate. Subsequently the license plate is progressed through Character Segmentation and Pattern Recognition to extract the data for manipulating and computing using database to manage the parking lot system efficiently. The system was experimented on 20 different license plates and the algorithms worked correctly in all cases with a very good response time.

Acknowledgment

First and foremost, we have to thank our research supervisor Rubel Biswas and co-advisor Zahangir Alam. Without their assistance and dedicated involvement in every step throughout the process, this paper would have never been accomplished. We would like to thank you very much for your support and understanding over the past year. I also wish to acknowledge our chairperson, Professor Mohammad Zahidur Rahman for his direct or indirect support and inspiration.

We also want to express our gratitude to all the people who have given their heart whelming full support in making this compilation a magnificent experience.

We also wanted to thank our family who inspired, encouraged and fully supported us for every trial that came our way, who helped us not only financially, but morally and spiritually.

Finally, we are very thankful to BRAC University, Bangladesh for giving us a chance to complete our B.Sc. degree in Computer Science and Engineering.

Supervised By

………………

List of Figures

Abbreviation

RGB -	Red Green Blue
OCR -	Optical Character Recognition
LPR –	License Plate Recognition

1. Introduction

Human beings are capable of identifying and recognizing patterns in an image but if these tasks are done in repetitive manner they are subjected to errors. The same happens in vehicle license plate recognition, especially because the number of vehicles is very large. That is why automated systems are required for this job. The objective is to extract and recognize vehicle registration numbers from car images, process the image data finally utilize for access record and then use it for further necessity. Monitoring the vehicle traffic and the management of parking areas are most labor-intensive job. Therefore the on systematic full automatic parking system is proposed. It differs from conventional parking system, no magnetic card is used to record the entry and exit time of. Also it is designed in a way that it has the ability of giving out the information regarding parking free spaces to users before entering the parking spaces. The car license plate recognition identification is an important application in the field of Intelligent Transport System (ITS) and Electronic toll collection (ETC). The objective is to extract and recognize vehicle registration numbers from car images, process the image data finally utilize for access record and prepare electronic bill. Electronic toll collection (ETC) or Electronic Car parking payment is one of the major research topics in intelligent transportation system (ITS) [1]. In the recent years, License Plate Recognition (LPR) are having a wide impact in people's life as their scope is to improve transportation safety and mobility and to enhance productivity through the use of advanced technologies. Productivity through the use of advanced technologies .This system is useful in many fields and places such as parking lots, private and public entrances, border control, theft and vandalism control. For the past two decades, there have been various studies for number plate recognition in images[2].

Figure (1.1): Parking Lot

Early day's number plate recognition systems employed detection methods based on techniques such as edge analysis[1,3,4], color analysis[5-6], and the Adaboost training [7]. S.Z. Wang and H.M. Lee[1], D. Zheng et al. [3], and P. Rattanathammawat and T.H. Chalidabhongse[8] proposed an edge analysis method for license plate detection. Ernst[7-9] introduced a face detection method based on local structure patterns computed by the Modified Census Transform (MCT).

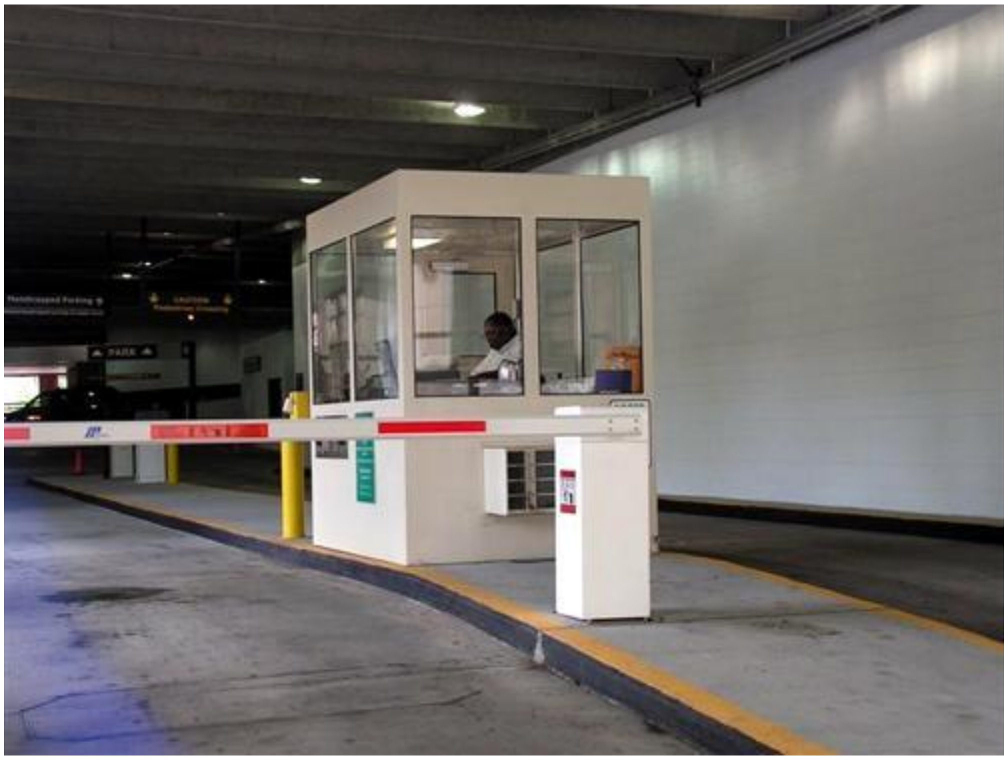


Figure (1.2): Booth

License plate recognition applies image processing and character recognition technology to identify vehicles by automatically reading the license plates. Basically, to build this system it consists of some major parts which are vehicle number plate extraction, characters segmentation, and characters recognition[11-14]. Vehicle number plates are extracted from the car plate images. Before extracting the number plate, the captured vehicle image should have been converted into grayscale format. After extracting the number plate, the characters are segmented using vertical and localization on the binary image. Optical Character Recognition (OCR) algorithm is used to recognize the character with condition[15], the background of the image has no or very little noise. The aim of the system is to recognize the license plate number of car from live video feed and store the data and time and generate electronic toll based on time spent.

1.1 Constraints

The system works with a few constraints.

- The camera is placed in a stationary position
- The parking lot area is brightly lit
- The car stops for a few second before getting assigned to a place
- No disturbing element in the background

1.2 Proposed System

The aim of our thesis is to provide automated parking lot management system for Bengali license plates using several algorithms such as Canny Edge Detection, Connected Component Labelling, Thresholding, Hough Transformation, Image Segmentation, OCR etc. There are mainly five parts in our thesis,

(1) Image Acquisition
(2) Preprocessing
(3) License Plate Localization
(4) Characters Segmentation
(5) Plate Number Recognition
(6) Data manipulation and Computations

The image is first acquired from a live video feed by comparing frames for significant changes. The image is then preprocessed for license plate recognition. The system of automatic number plate recognition faces many challenges. So, preprocessing is essential to enhance the input and making it more suitable for the next processing steps. The first step done in the preprocessing is to convert the image to grayscale and detect the edges using canny edge detection. Finally, the connected components are labelled and separated for license plate localization.

In the localization stage, the license plates are located by using the hough transformation method and output the sub-image that contains the license plate. The characters are then segmented in the Characters Segmentation phase. This is done by filtering the black and white color of the binary picture.
Finally, the characters are matched against sample Bengali characters via OCR and the license plate number is generated. The number is then stored in the database along with the time stamp.

1.3 Thesis Outline

The structure of the paper is as follows. Section 2 presents the related work as my system and the
drawbacks. The algorithms, detailed procedures of how I applied the algorithms and designed my system are mentioned in section 3. In section 4, the result and analysis is given. Finally the conclusion and future work is mentioned in section 5.

2. Background Analysis

2.1. Previous Work

Even though there are some professional software available for automated car parking lot management, there are only a few papers published on automated car parking lot management. IAITO INFOTECH TECHNOLOGIES used RFID based automated car parking lot management. It uses radio-frequency identification system uses intelligent bar codes to track cars[16]. There has also been the use of image processing system which includes morphological operations with good thresholds, normalization, OCR to detect license plates[2]. There has been no works on Bengali license plate based automatic car parking lot management.

2.2. Automatic License Plate Detection

Though the applications of automatic license plate detection have emerged in the last decade or so, the technology has been present for nearly 30 years. In the late 1970s, researchers for the United Kingdom police department manufactured the first working license plate recognition system and began deploying it by the beginning of the 1980s[17]. The first US patent for an automatic license plate reader was issued in 1989[18]. In a rare break for computer science, some issuing states/countries around the world adapted their license plates to assist automatic license plate recognition systems. In 2003[17], the Netherlands modified their license plate by introducing a new a typeset font. During the same year, Texas passed a bill banning novelty frames (later overturned in 2007) because they impeded the view of the license plate to recognition systems[18]. There has been 2 works on Bengali License Plate Detection so far. One of them involves the use of ROID algorithm to localize the license plate[19] while the other uses segmentation and recognition[20].

2.3. Optical Character Recognition

Optical character recognition (OCR) is the electronic conversion of handwritten, typewritten, or printed text from still or motion images to machine-encoded text. Common uses include scanning of books for electronic retrieval or scanning to edit documents electronically. OCR dates back to the late 1920s when Gustav Tauschek first obtained a patent in Germany[21]. Today, machine printed OCR is regarded as a “solved problem.” Researchers are challenged to conduct OCR on complex images which include complicated backgrounds, degradation, low resolution, rotation and such. Many of these issues are what make license plate recognition quite challenging. Hand-printed and cursive writing, and East Asian and Latin-script characters are also regarded as complex tasks in OCR. The first Bengali OCR software was released on 2014[22]. There has been the use of tesseract to develop Bengali OCR system[23],[24].

3. System Design

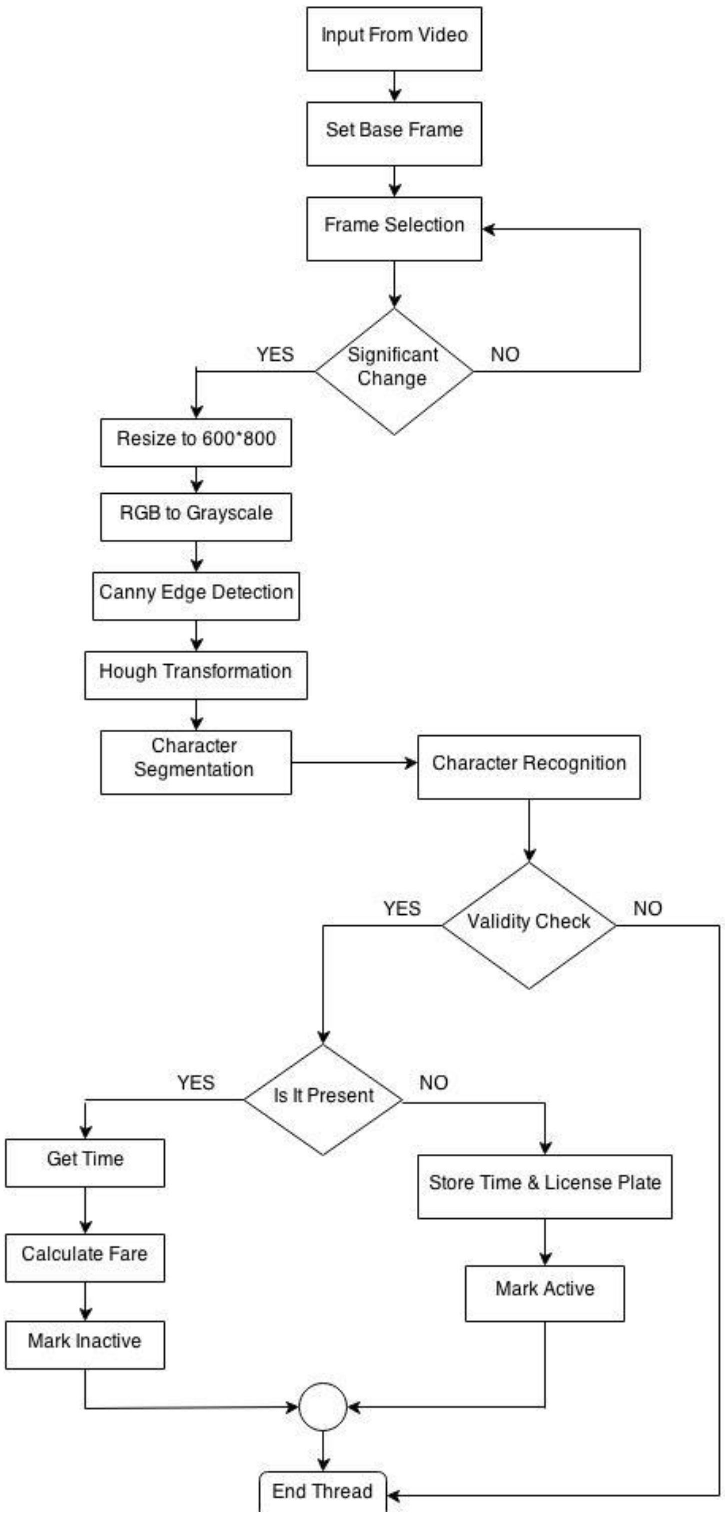


Figure (2.1): System Design

3.1. Image Acquisition

Image acquisition is the process of obtaining an image from the camera. This is the first step of any vision based systems. In our current research a live video camera is placed in a stationary position with ample amount of light. The system extracts frame every second and count pixels. It compares with the previous frame for pixel change. If there is no dramatic change in pixels and the pixel count is not equal to the defined blank, the system sends the picture for license plate recognition.

3.2. Preprocessing

The preprocessing is used to enhance the image and make it suitable license plate localization. The preprocessing is done in several methods.

3.2.1. Resizing the Image

After the image is acquired, the image is resized to 800 x 600 resolution. The resizing is used to make the image clearer and more efficient for detecting the license plate[2].

3.2.2. Convert Image to Grayscale

Many of the systems that capture license plates do so in the near-infrared spectrum. Therefore, the image is first converted from color to gray scale. This was also necessary to create a binary image of the input source.
Image of color to Gray scale. The RGB image is converted to gray scale by eliminating the hue and saturation information while retaining the luminance. This is done by converting RGB values to gray scale values by forming a weighted sum of the R, G, and B components[25]:

$$0.2989 * R + 0.5870 * G + 0.1140 * B$$

3.3. License Plate Localization

3.3.1. Edge Detection

Edge detection is the name for a set of mathematical methods which aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed edges. There are several types of edge detection algorithms[27],

- Sobel
- Prewitt
- Canny

Edge detection is used to distinguish the edges in a picture which will help us localize the position of the license plate. After experimenting with different types of edge detection methods, canny edge detection provided the best result.

Canny edge detection is a four step process.

1. A Gaussian blur is applied to clear any speckles and free the image of noise.
2. A gradient operator is applied for obtaining the gradients' intensity and direction.
3. Non-maximum suppression determines if the pixel is a better candidate for an edge than its neighbors.
4. Hysteresis thresholding finds where edges begin and end.

3.3.1.1. Noise reduction

The image after a 5x5 Gaussian mask has been passed across each pixel.

Because the Canny edge detector is susceptible to noise present in raw unprocessed image data, it uses a filter based on a Gaussian (bell) curve, where the raw image is convolved with a Gaussian filter. The result is a slightly blurred version of the original which is not affected by a single noisy pixel to any significant degree.

Here is an example of a 5x5 Gaussian filter, used to create the image to the right, with σ = 1.4. (The asterisk denotes a convolution operation.)

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}.$$

3.3.1.2. Finding the intensity gradient of the image

An edge in an image may point in a variety of directions, so the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. The edge detection operator (Roberts, Prewitt, Sobel for example) returns a value for the first derivative in the horizontal direction (G_x) and the vertical direction (G_y). From this the edge gradient and direction can be determined:

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$$
$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

Where G can be computed using the hypot function and atan2 is the arctangent function with two arguments. The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0, 45, 90 and 135 degrees for example).
Non-maximum suppression

3.3.1.3. Non-maximum suppression is an edge thinning technique

Given estimates of the image gradients, a search is carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction. In some implementations, the algorithm categorizes the continuous gradient directions into a small set of discrete directions, and then moves a 3x3 filter over the output of the previous step (that is, the edge strength and gradient directions). At every pixel, it suppresses the edge strength of the center pixel (by setting its value to 0) if its magnitude is not greater than the magnitude of the two neighbors in the gradient direction. For example,

- if the rounded gradient angle is zero degrees (i.e. the edge is in the north–south direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the east and west directions,
- if the rounded gradient angle is 90 degrees (i.e. the edge is in the east–west direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the north and south directions,

- if the rounded gradient angle is 135 degrees (i.e. the edge is in the northeast–southwest direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the north west and south east directions,
- if the rounded gradient angle is 45 degrees (i.e. the edge is in the north west–south east direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the north east and south west directions.

In more accurate implementations, linear interpolation is used between the two neighbouring pixels that straddle the gradient direction. For example, if the gradient angle is between 45 degrees and 90 degrees, interpolation between gradients at the north and north east pixels will give one interpolated value, and interpolation between the south and south west pixels will give the other (using the conventions of last paragraph). The gradient magnitude at the central pixel must be greater than both of these for it to be marked as an edge.

Note that the sign of the direction is irrelevant, i.e. north–south is the same as south–north and so on.

3.3.1.4. Tracing edges through the image and hysteresis thresholding

Large intensity gradients are more likely to correspond to edges than small intensity gradients. It is in most cases impossible to specify a threshold at which a given intensity gradient switches from corresponding to an edge into not doing so. Therefore Canny uses thresholding with hysteresis.

Thresholding with hysteresis requires two thresholds – high and low. Making the assumption that important edges should be along continuous curves in the image allows us to follow a faint section of a given line and to discard a few noisy pixels that do not constitute a line but have produced large gradients. Therefore we begin by applying a high threshold. This marks out the edges we can be fairly sure are genuine. Starting from these, using the directional information derived earlier, edges can be traced through the image. While tracing an edge, we apply the lower threshold, allowing us to trace faint sections of edges as long as we find a starting point.

Once this process is complete we have a binary image where each pixel is marked as either an edge pixel or a non-edge pixel. From complementary output from the edge tracing step, the binary edge map obtained in this way can also be treated as a set of edge curves, which after further processing can be represented as polygons in the image domain[28].

3.3.2. Connected Component Labeling

Connected-component labeling is an algorithmic application of graph theory, where subsets of connected components are uniquely labeled based on a given heuristic. A graph, containing vertices and connecting edges, is constructed from relevant input data. The vertices contain information required by the comparison heuristic, while the edges indicate connected 'neighbors'. An algorithm traverses the graph, labeling the vertices based on the connectivity and relative values of their neighbors. Connectivity is determined by the medium; image graphs, for example, can be 4-connected or 8-connected[29].

1. 4-connected: 4-Connected pixels are the neighbors to every pixel that touches on one of their edges. These pixels are connected both horizontally and vertically. In the terms of pixel coordinates, every pixel having the coordinates or is connected to the pixel at(x,y)[30].
2. 8-connected: 8-connected pixels are the neighbors to every pixel that touches on one of their edges or the corners. These pixels can be connected horizontally, vertically, and diagonally. In addition to the 4-Connected pixels, each pixel with the coordinates or is connected to pixel at(x,y)[30]. The connected Components are then found out in the filtered image and each connected component is labeled.

Following the labeling stage, the graph may be partitioned into subsets, after which the original information can be recovered and processed.

We used 8-connected algorithm to process the image and find connected components. The system uses the general procedure outlined in reference[31], pp. 40-48:

1. Run-length encode the input image.
2. Scan the runs, assigning preliminary labels and recording label equivalences in a local equivalence table.
3. Resolve the equivalence classes.
4. Relabel the runs based on the resolved equivalence classes.

3.3.3. Bounding Box

The minimum or the smallest bounding or the enclosing box for any point set in N dimensions is the box with the smallest measure (of area, volume, or hypervolume in higher dimensions) within which all points lie. When the other kinds of measure are used, the minimum box is usually called accordingly depending on the measure used, e.g., "minimum-perimeter bounding box". The minimum bounding box of any point set is same as the minimum bounding box of its convex hull, this is a fact which can be used heuristically to speed up computation. The term "box"/"hyperrectangle" has come from its usage in Cartesian coordinate system, where it can be indeed visualized as a rectangle (in two-dimensional case), rectangular parallelepiped (in case of three-dimensional), etc In the case of two-dimensional it is called the minimum bounding rectangle[32]. In other words it is a rectangle which has the minimum height and width that covers all the pixels present in a particular connected component or region[32].

3.3.4. Hough Transformation

3.3.4.1. Hough Transformation Theory

In automated analysis of digital images, a sub problem often arises of detecting simple shapes, such as straight lines, circles or ellipses. In many cases an edge detector can be used as a pre-processing stage to obtain image points or image pixels that are on the desired curve in the image space. Due to imperfections in either the image data or the edge detector, however, there may be missing points or pixels on the desired curves as well as spatial deviations between the ideal line/circle/ellipse and the noisy edge points as they are obtained from the edge detector. For these reasons, it is often non-trivial to group the extracted edge features to an appropriate set of lines, circles or ellipses. The purpose of the Hough transform is to address this problem by making it possible to perform groupings of edge points into object candidates by performing an explicit voting procedure over a set of parameterized image objects (Shapiro and Stockman, 304).

The simplest case of Hough transform is the linear transform for detecting straight lines. In the image space, the straight line can be described as $y = mx + b$ where the parameter m is the slope of the line, and b is the intercept (y-intercept). This is called the slope-intercept model of a straight line. In the Hough transform, a main idea is to consider the characteristics of the straight line not as discrete image points (x1, y1), (x2, y2), etc., but instead, in terms of its parameters according to the slope-intercept model, i.e., the slope parameter m and the intercept parameter b. In general, the straight line $y = mx + b$ can be represented as a point (b, m) in the parameter space. However, vertical lines pose a problem. They are more naturally described as $x = a$ and would give rise to unbounded values of the slope parameter m. Thus, for computational reasons, Duda and Hart proposed the use of a different pair of parameters, denoted r and \theta (theta), for the lines in the Hough transform. These two values, taken in conjunction, define a polar coordinate.

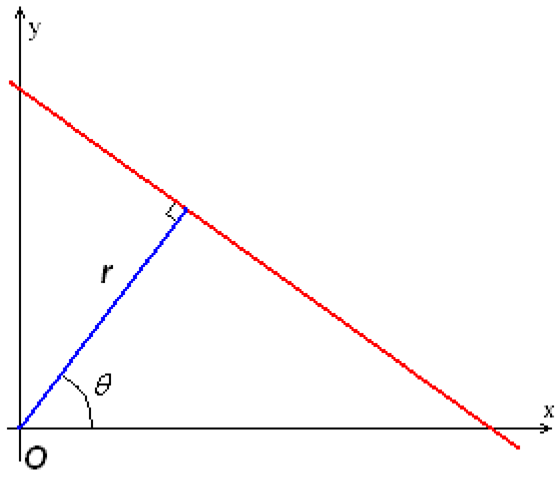


Figure (3.1): Hough Transformation

The parameter r represents the algebraic distance between the line and the origin, while θ is the angle of the vector orthogonal to the line and pointing toward the half upper plane. If the line is located above the origin, θ is simply the angle of the vector from the origin to this closest point. Using this parameterization, the equation of the line can be written as:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$

This can be rearranged to:

$$r = x\cos\theta + y\sin\theta$$

It is therefore possible to associate with each line of the image a pair (r,θ) which is unique if $\theta \in [0,\pi)$ and $r \in \mathbf{R}$, or if $\theta \in [0,2\pi)$, or if $\theta \in [0,2\pi)$ and $r \geq 0$. The (r,θ) plane is sometimes referred to as Hough space for the set of straight lines in two dimensions. This representation makes the Hough transform conceptually very close to the two-dimensional Radon transform.

For an arbitrary point on the image plane with coordinates, e.g., (x0, y0), the lines that go through it are the pairs (r,θ) with

$$r(\theta) = x_0 \cos\theta + y_0 \sin\theta,$$

Where r (the algebraic distance between the line and the origin) is determined by $\theta \in [0,\pi)$.
If r is required to be positive, then θ must vary in $[0,2\pi)$. In other words, θ is the angle of the vector from the origin and this closest point (if $r \neq 0$), or the angle of the vector orthogonal to the line and pointing to the half upper plane (if r = 0). The lines that go through (x0, y0) are then

$$r(\theta) = |x_0 \cos\theta + y_0 \sin\theta|.$$

These representations corresponds to a sinusoidal curve in the (r,θ) plane, which is unique to that point. If the curves corresponding to two points are superimposed, the location (in the Hough space) where they cross corresponds to a line (in the original image space) that passes through both points. More generally, a set of points that form a straight line will produce sinusoids which cross at the parameters for that line. Thus, the problem of detecting collinear points can be converted to the problem of finding concurrent curves[33].

3.3.4.2. Modified Hough Transformation

3.3.4.2.1. Concept

The MATLAB Hough Transformation method computes the Standard Hough Transform (SHT) of a binary image. It uses the Hough function to detect lines in an image. The function returns H, the Hough transform matrix. theta (in degrees) and rho are the arrays of *rho* and *theta* values over which Hough generates the Hough transform matrix. The image can be logical or numeric, and it must be real, 2-D, and nonsparse.

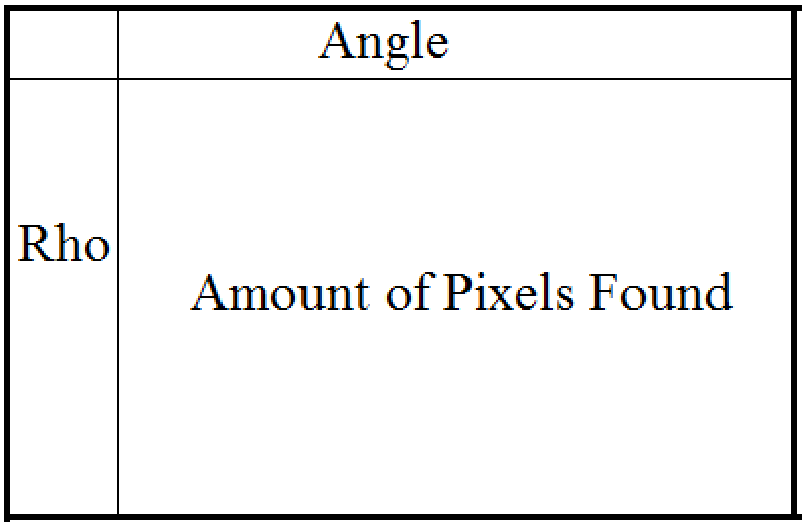


Figure (3.2): Hough Transformation Matrix

The Hough transformation method in MATLAB gives us a 2D matrix containing Angle and Rho pixels. Instead of using all the angles we have used only two angles which are ZERO and NINETY degrees. We will be working only on these two angles and ignore the rest as our work is on only rectangular area. It minimizes a lot of extra processes and optimizes the system. According to our concept, for a perfect rectangle Hough transformation gives us a figure like this:

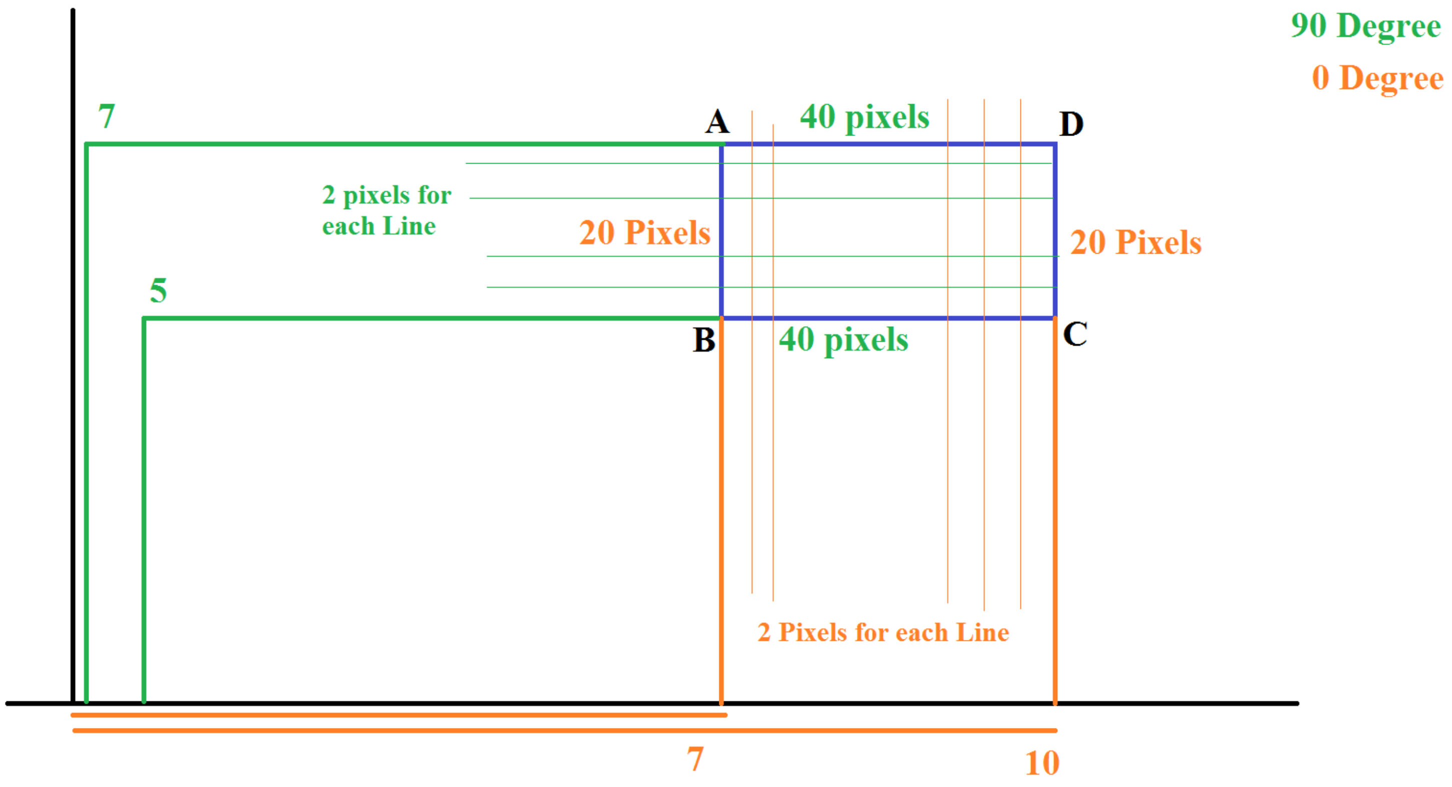
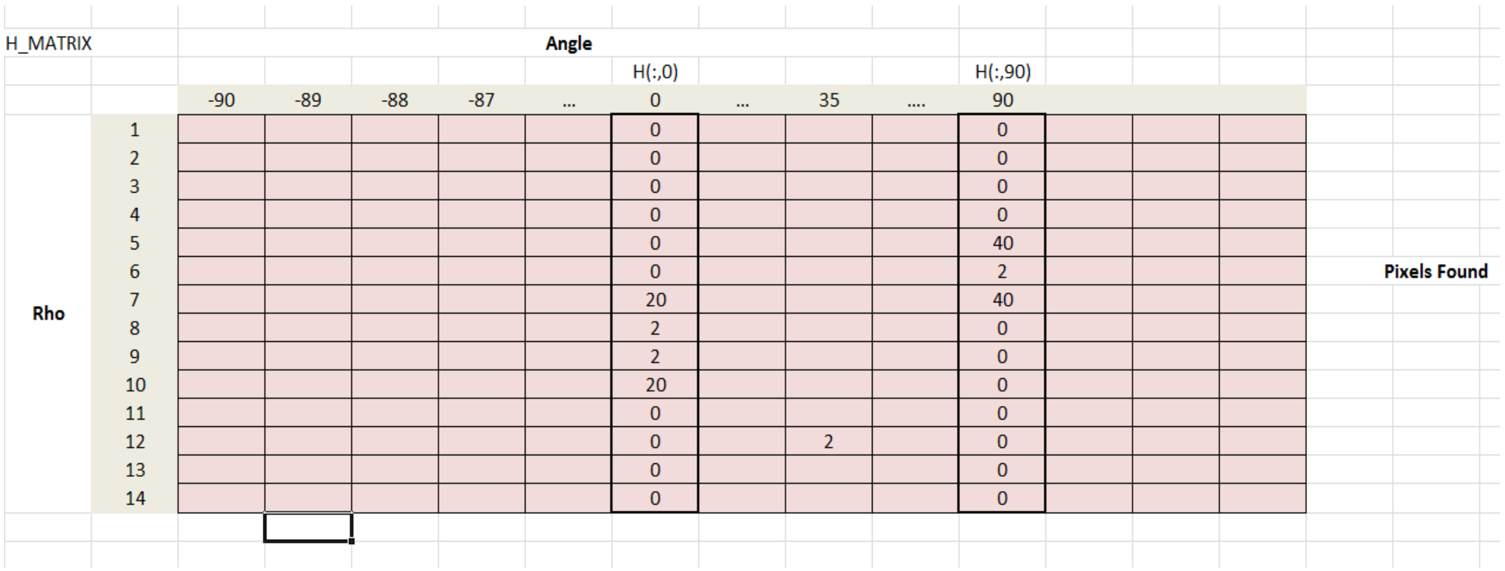


Figure (3.3): Perfect Rectangle.

In the figure, X axis indicates the values of Rho. For example, if we consider ZERO degree angle, it is showing that, the rho values are 7 and 10 where the orange line creates an angle of zero with the X-axis. From the x's value 0 to 7, it gives us zero pixels as there are no pixels crossed drawing perpendicular lines on the x's value 0 and 7. When the x's value is 7, it gives us the highest amount of pixel values as the crossing points are at a higher rate, drawing a perpendicular line on the point (7,0), which is 20 pixels assuming. Then again, from the x's value 7 to 10 we find only two pixels as the perpendicular lines only gives us two cross points for the rectangle. When the x's value increases to 10, it again gives us a high rate of amount of pixels crossed, drawing a perpendicular line, assuming it to be 20 pixels. The latest value which we found as the value of the matrix are the two sides of the rectangle, BC and BD from the figure(). On the other hand, if we consider Ninety degree where rho is ninety degree angled with the x-axis, indicating the green line, direct perpendicular lines are drawn. Assuming the values at which found the highest cross points are at 5 and 7, carrying 40 pixels. Between these two points all the perpendicular lines crosses only two points as the concentration is very low. So, in this way we find two pixel values for each angles. For the second case assuming the values are found for the lines AB and CD from the rectangle. Calculating and verifying those values we can determine the desired rectangle.

Obviously this is for the perfect rectangle’s case. Below a figure is given, showing a possible matrix for a perfect rectangle.



H_MATRIX						Angle									
							H(:,0)				H(:,90)				
		-90	-89	-88	-87	...	0	...	35	....	90				
	1						0				0				
	2						0				0				
	3						0				0				
	4						0				0				
	5						0				40				
	6						0				2				Pixels Found
Rho	7						20				40				
	8						2				0				
	9						2				0				
	10						20				0				
	11						0				0				
	12						0		2		0				
	13						0				0				
	14						0				0				

Figure (3.4): Hough Transformation Matrix – Sample

In the figure, for zero degree angles, it gives 20 pixels at 7 and 10, which is the value of rho. Between these values, the amount of pixels crossed is only two. It is similar for the case of ninety degree also.

But in real life situation, we might not exactly get the perfect rectangle. As the lines of the rectangles are not in a straight line, it is likely to have several amount of high density of pixel crossed in different rho values. An image is shown below:

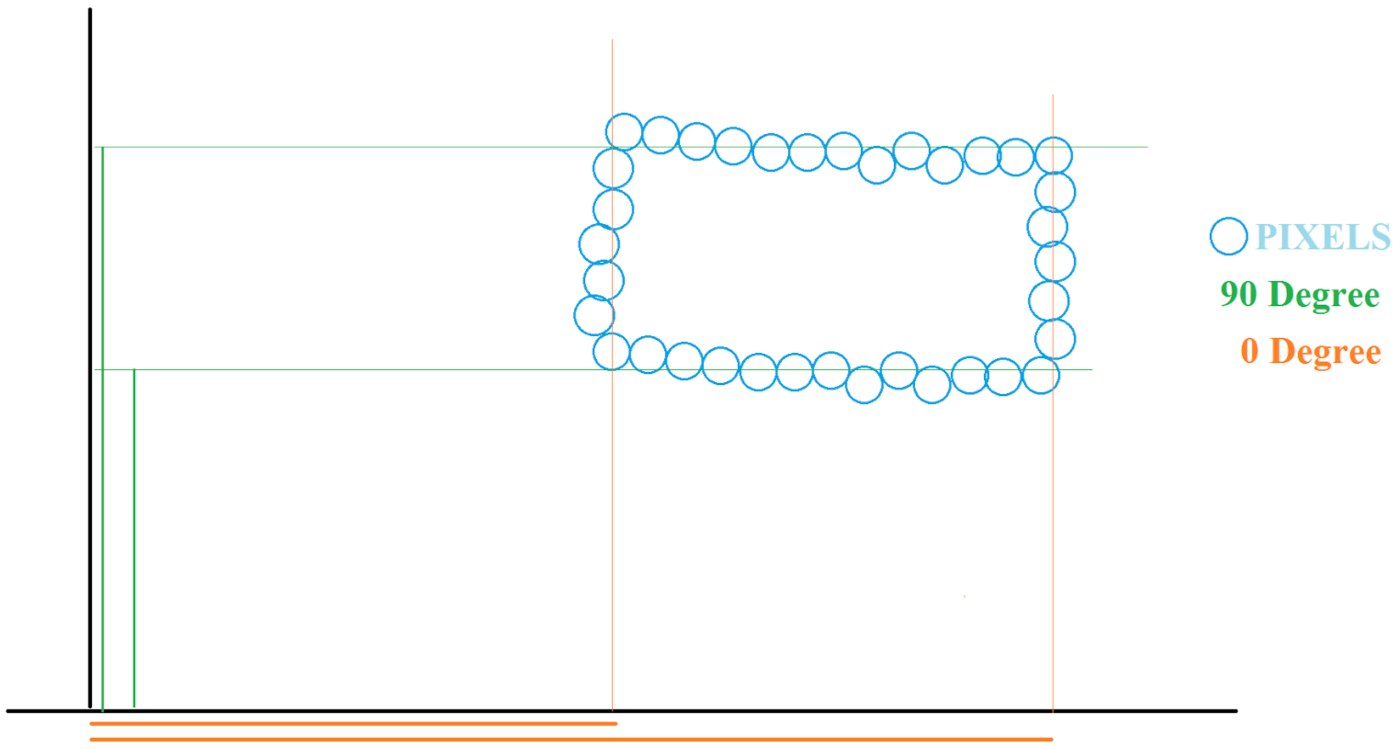


Figure (3.5): Sample of License plate in real

In the above figure, it is can be labeled as an example of real life license plate. The four sides of the rectangle may not be in a straight line rather it can be scattered like the above figure (3.5). Here, instead of having exactly one rho value for a line, indicating a perfect high density of amount of pixels crossed, we are likely to find more than one rho values showing more than two pixels crossed points. So, for several rho values we get several amounts of crossed values of pixels.

In this way a sample figure can be found from the connected component which is shown below:

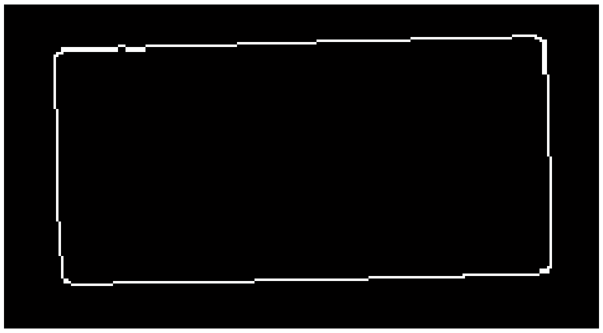


Figure (3.6): License Plate From Connected Component

If we zoom it up a little bit, we find the below picture of a particular car’s license plate:

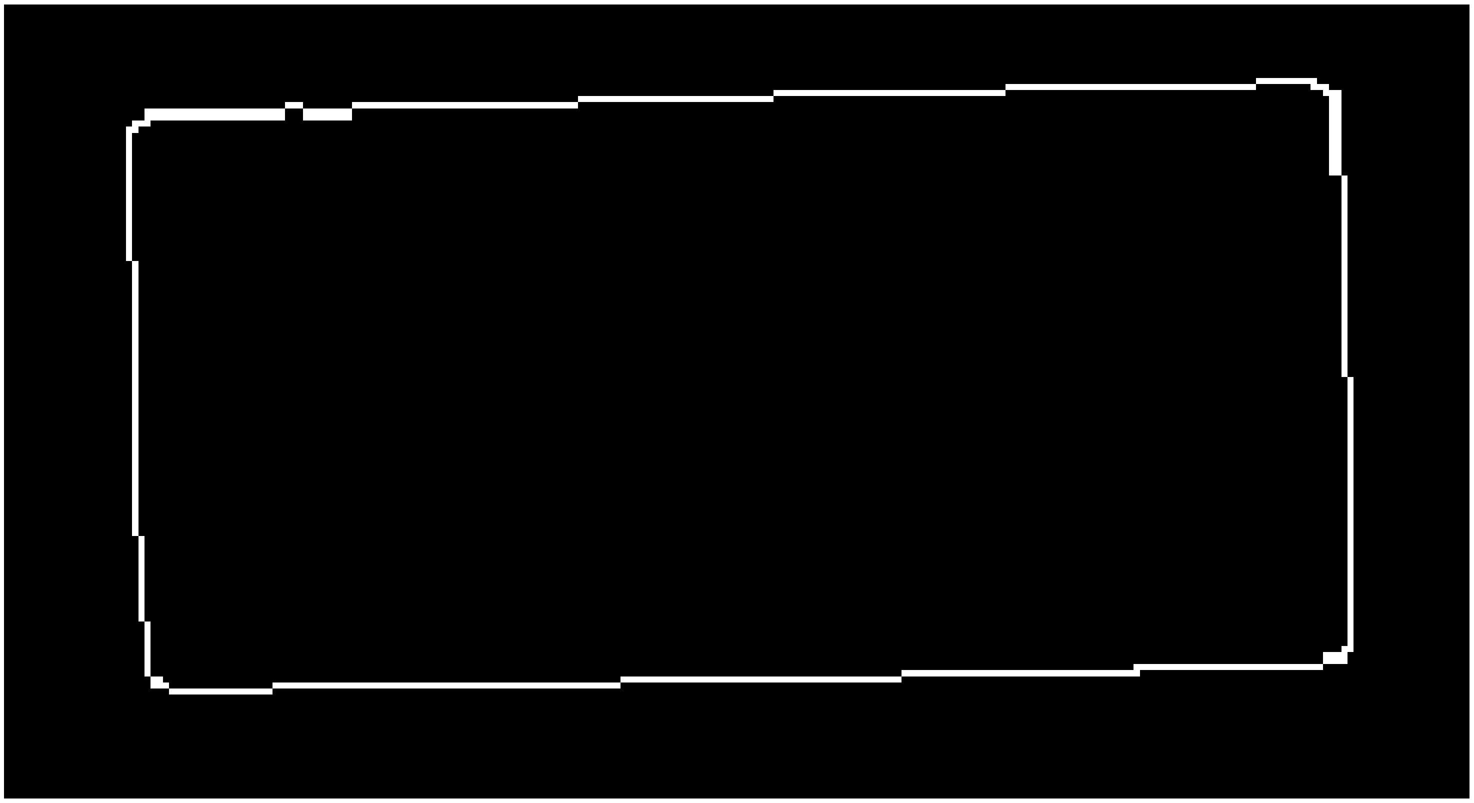


Figure (3.7): License Plate (Zoomed).

Here, it is clear that, we not getting the perfect four lines for the rectangle as the line’s pixels are dispersed. Our previous discussion gives a clear concept to do what in this situation.

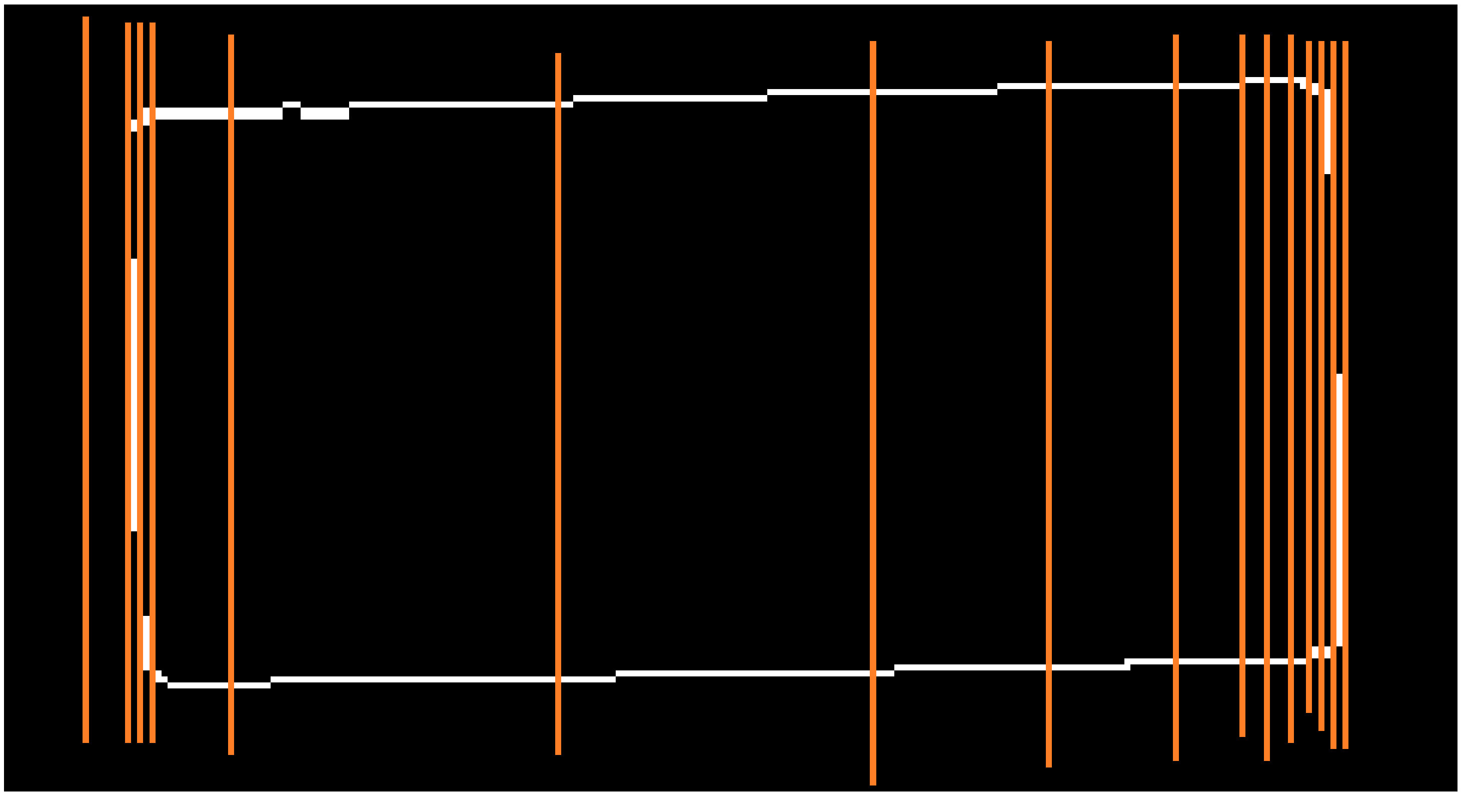


Figure (3.8): Concept for Zero Degree Angle

To make it more clearer, the above image is showing the perpendicular lines that are drawn for the angle of zero degree as the rho values are on x axis, making zero degree with the x-axis. At first, there is no pixel found drawing the perpendicular lines. But as we increase our rho value and the perpendicular line from those rho values are likely to cross pixels on the way. When the value of rho increase, for a certain moment it will find only two pixels crossed, increasing the value more and more, we end up finding the other end of high density of crossed pixels which is actually the opposite line of the first found line.

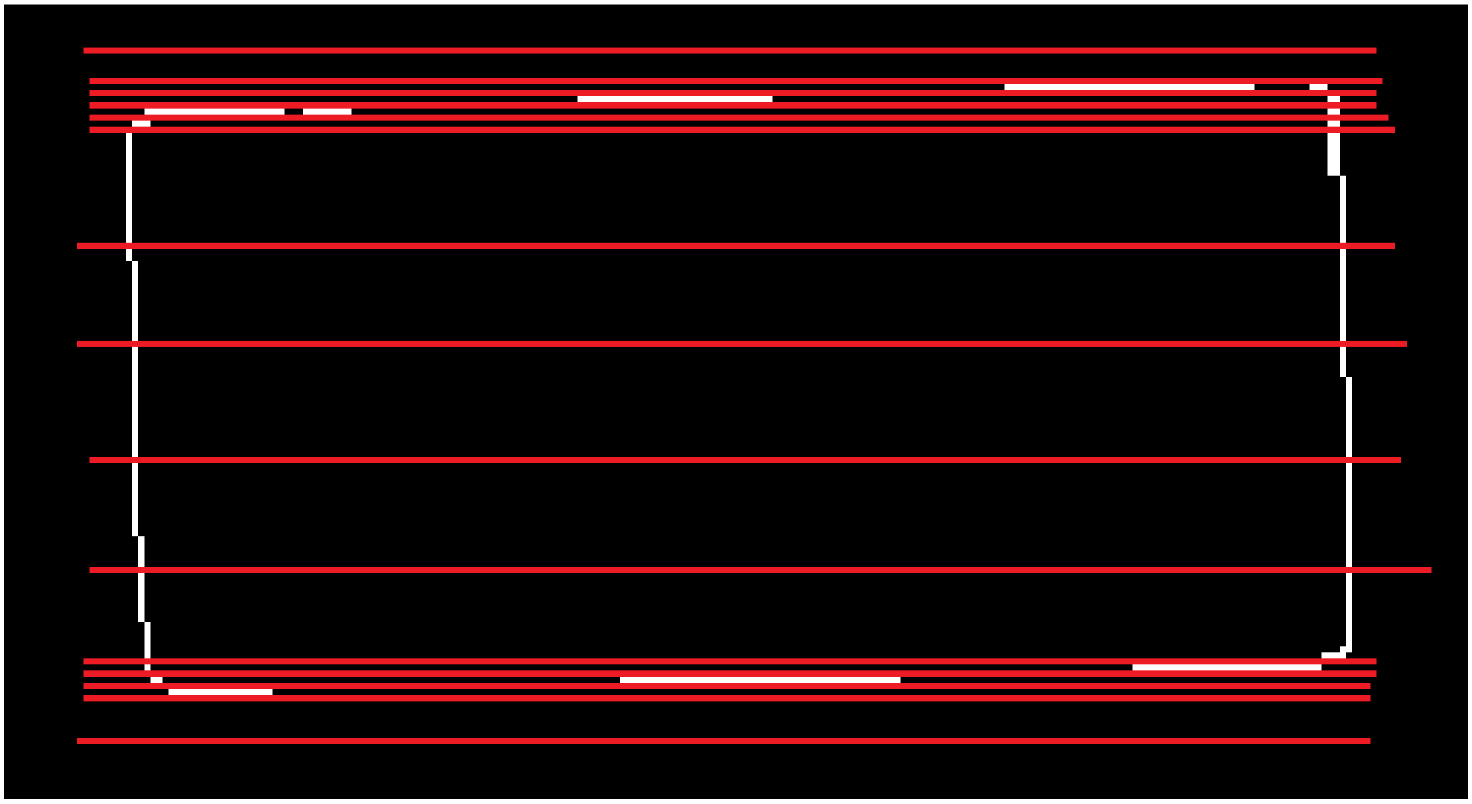
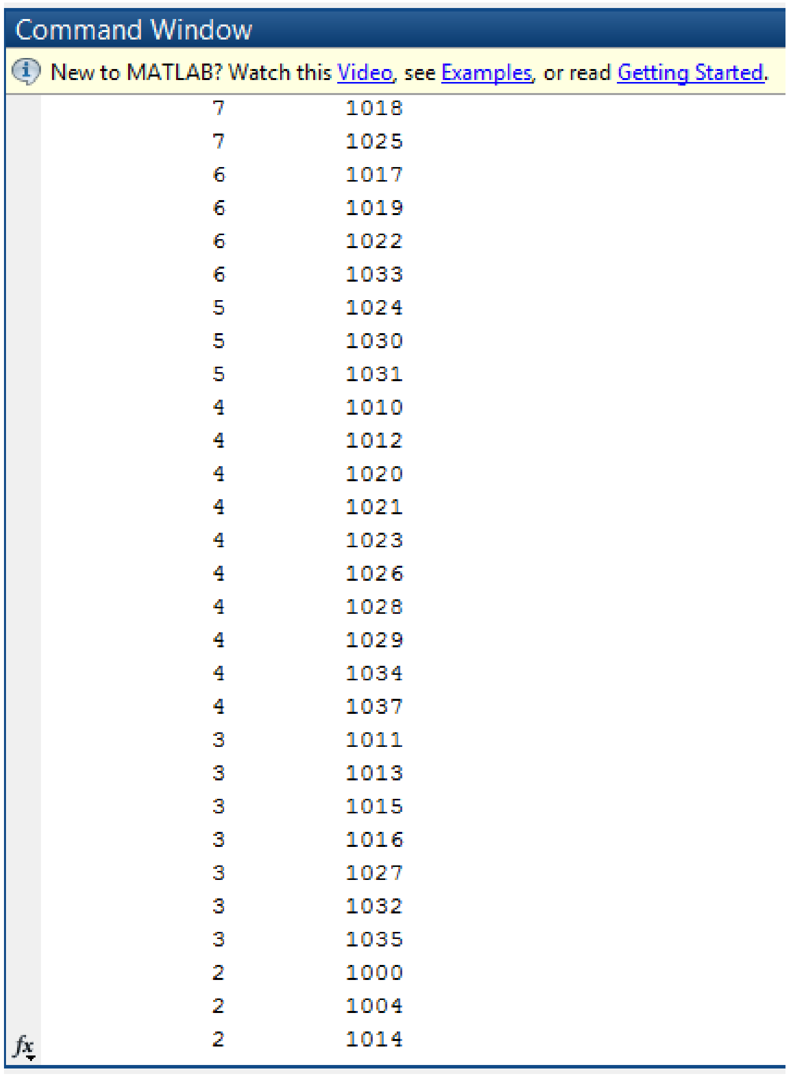


Figure (3.9): Concept for Ninety Degree Angle.

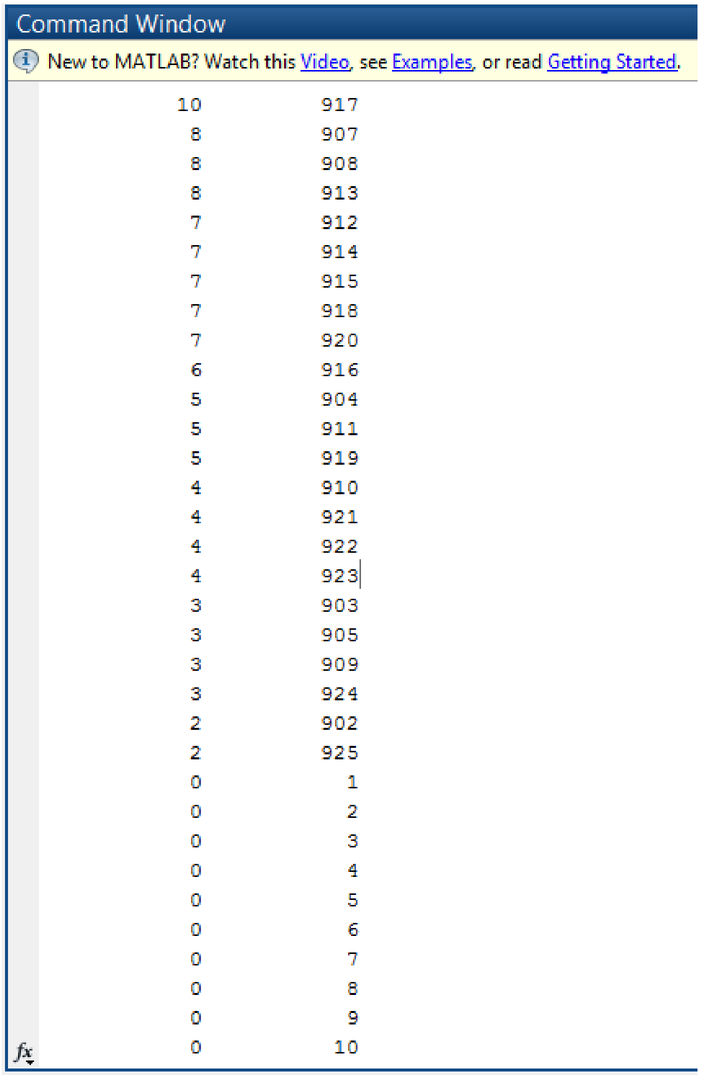
Here, the above image is showing the perpendicular lines drawn for the angle of ninety degree as the rho values are on y-axis and they are itself perpendicular with the x-axis. Drawing perpendicular lines from several rho values we again find following by no pixels, high density, two pixels, high density and no pixels crossed again. Calculating these crossed pixel values will lead us to a conclusion of the length measurement.

If we run the above rectangle license plate on our own Hough Transformation algorithm, the 2D matrix can be something of this:


Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
     7        1018
     7        1025
     6        1017
     6        1019
     6        1022
     6        1033
     5        1024
     5        1030
     5        1031
     4        1010
     4        1012
     4        1020
     4        1021
     4        1023
     4        1026
     4        1028
     4        1029
     4        1034
     4        1037
     3        1011
     3        1013
     3        1015
     3        1016
     3        1027
     3        1032
     3        1035
     2        1000
     2        1004
     2        1014
```

Figure (3.10): Matrix for Zero Degree


Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
    10         917
     8         907
     8         908
     8         913
     7         912
     7         914
     7         915
     7         918
     7         920
     6         916
     5         904
     5         911
     5         919
     4         910
     4         921
     4         922
     4         923
     3         903
     3         905
     3         909
     3         924
     2         902
     2         925
     0           1
     0           2
     0           3
     0           4
     0           5
     0           6
     0           7
     0           8
     0           9
     0          10
```

Figure (3.11): Matrix for Ninety Degree

In the figure() and figure(), the values of rhos and pixels found are sorted in descending order. The first column carrying the values of rho and the second column carrying the values of the amount of pixels found in total. To analyze it we can say that, for the above license plate from figure (3.11), our algorithm found 917 pixels on 10 value of rho, 907 pixels on 8 value of rho and so on.

These values can be plotted on graph as below:

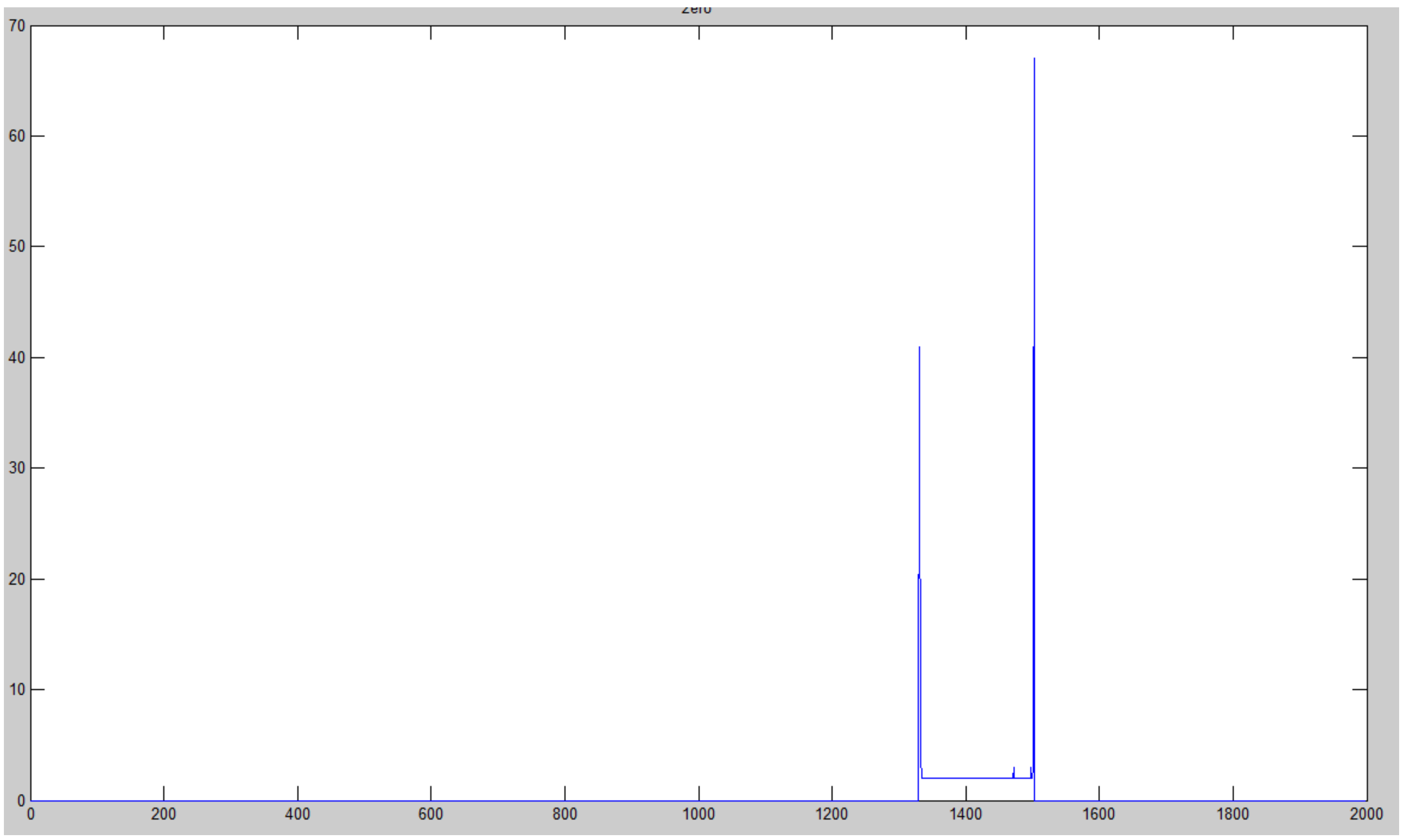


Figure (3.12): Plotted Graph from Individual Matrix

Here, in the figure (3.12), along with X-axis, it is the value of Rho and along with Y-axis, it the value of pixels found. The graph is indicating that, along with the x-axis, at first the value of y (amount pixels found) is zero as no pixels found so far. But when the x's value reaches to approximately 1350, it found a great density of pixels. As we go further the peak point rises indicating that the density is increasing of the crossed pixels. Then again it drops to the value of 2 on y's axis indicating that it is scanning in the middle part of the rectangle. Similarly, algorithm gets the second line which is approximately on 1550 of x's value. If we calculate the two high pick point values, we are likely to find out a length of a line.

3.3.4.2.2. Rules

The set of rules to detect rectangle in any image are as follows:

We must use some threshold values for checking the rectangle we are considering. In order to do so, we used some basic mathematical terms.

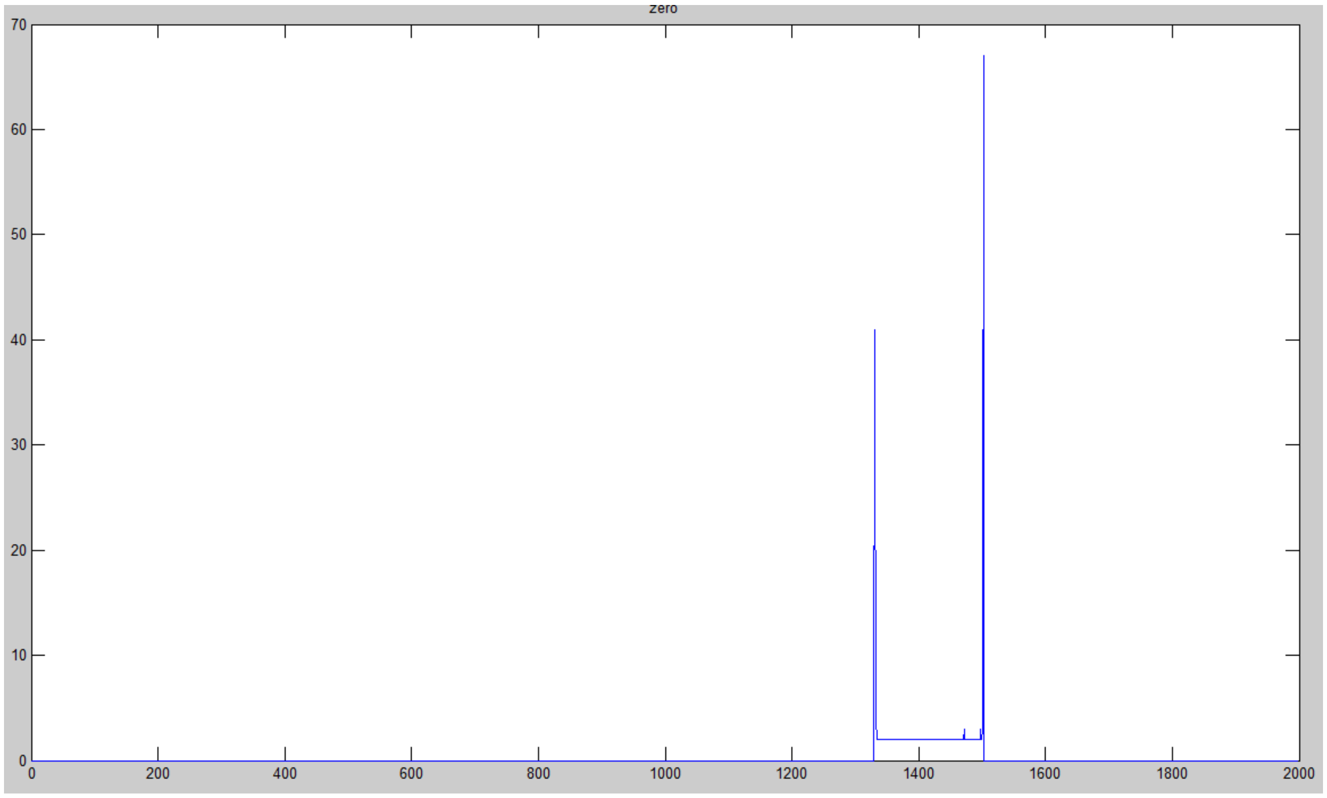


Figure (3.13): Rules

Rule 1

In the graph above, for the values of Y (which is actually the value of amount of pixels found) we are considering only the 10% more or less of the first high peak point. The mathematical term is as following:

$$T = 1.1 \times \frac{\sum_{i=1}^{n} Hs_i}{n}$$

[Where Hs = all the amount of pixels found at peak]

Rule 2

As for the x's value, we do not want a scattered collection values, because to much scattered may mean that there are more than two straight lines. The mathematical term for this threshold is as below :

$$T = |x_i - x_j|$$

We fixed a constant value 8 for this threshold T. The difference of two x's values indicating more than 8 will consider that there are more than two lines for a particular angle.

Rule 3

After finishing up the algorithm, when we find the two correspondence length of a rectangle, we now check the difference. The difference being farfetched will indicate that the two lines might not be the opposite of each other and they might not be the two particular line of a rectangle what we are desired to find. Here the mathematical term is in two sections as below:

$$T_1 = \frac{\sum_{i=1}^{n} line\_length_i}{n}$$

[The average value of the two lines]

$$T_2 = \frac{|Difference\ of\ the\ line\ length\ value\ |}{T_1} \times 100$$

[Percentage of the difference]

The percentage threshold value is considered 15%. So, if the two line's threshold value is above 15%, we are not considering it to be the line of two of a rectangle. It must be closer to 15%.

Rule 4

After having the two length of a rectangle what we desire to find, it is time to check the ratio. The ration we considered is only 3. It would be a hard check if the two lines are considerable or not.

$$Ratio = \frac{Line\ length\ value\ from\ 90\ degree\ angle}{Line\ length\ value\ from\ 0\ degree\ angle}$$

After verifying all the rules and if we get a positive result we can say that a rectangle is found and it is ready for the further processes.

3.3.5. Cropping the Localized License Plate

After identifying the best possible bounding box candidate for the license plate the coordinates of the bounding box are noted and the box is cropped from the image and sent to Character segmentation module for further processing.

3.4. Character Segmentation

3.4.1. Separating the Rows

The character segmentation process acts as a bridge between the license plate localization and optical character recognition modules. Its main function is to segment the characters in the selected candidate region (extracted license plate) such that each character can be sent to the optical character recognition module individually for recognition.

Once the license plate is localized we proceed to obtain the individual characters. Bengali License plates are divided into two rows. The first row contains the regional information of the license plate and the next row contains the plate number. A license plate has high intensity variation regions (due to alternating white and black regions).This forms the basis for character segmentation. By various observations we observed that for the license plate regions the amount of white (number or text) on black (background region) or vice versa, is specific for the number regions and falls within a certain range. We ignore those regions which are out of range to isolate the number regions[34]. Morpho-logical techniques are used to remove small white areas which escape range corrections (certain shadows or text which show similar patterns to numbers). Finally individual characters are extracted to pass on through the optical character recognition (OCR) system. Segmentation is one of the most important processes in the automatic number plate recognition, because all further steps rely on it. If the segmentation fails, a character can be improperly divided into two pieces, or two characters can be wronged merged together which would lead to the failure of following stages of recognition[35]. We can use a horizontal projection of a number plate for segmentation. Firstly, we take the binary version of the license plate and scan the rows for white and black pixel. The black pixels are cut from the picture and white pixel areas are separated. In this way, we will find 2 rows containing the regional and license number information from the license plate image.

3.4.2. Characters Segmentation

We can use a horizontal projection for each rows of a number plate for segmentation. We compute a horizontal projection px(x) of the plate f (x, y). We use this projection to determine horizontal boundaries between segmented characters[34] These boundaries correspond to peaks in the graph of the horizontal projection The goal of the segmentation algorithm is to find peaks, which correspond to the spaces between characters. At first, there is a need to define several important values in a graph of the horizontal projection px(x).

- V_m : The maximum value contained in the horizontal projection
- V_a : The average value of horizontal projection
- Max peak : The highest peak in the segmented plate image

The aim of this step is to find the boundaries between two consecutive characters on the number plate. By obtaining the horizontal projection we can consider those peaks which are greater than a certain threshold value as these boundaries. From the above figure we consider only those peaks which are greater than 0.9 times the max peak value as legitimate boundaries and hence obtain the coordinates of each character[36]. We take 0.9 the max peak value to account for noise which may have escaped corrections carried out in previous steps. The regions between consecutive peaks contain one character each which can be thereafter extracted from the plate region and sent for character recognition. The last step of the phase of character segmentation is to obtain the individual characters. This is fulfilled by using the above obtained boundary values. These individual characters are then fed to the next step of analysis which is the final step called the Optical character recognition.

Before the characters can be fed to the next step we have to remove redundant back-ground areas from the individual images of characters. The resultant images will consist of properly bounded numbers or alphabets of the license plate characters which will make our work of OCR easier. We do this by scanning each image first horizontally and then vertically to obtain the startrow, endrow, startcolumn and endcolumn which denote the two dimensional boundary values in which the numbers or alphabets fit perfectly[36].

3.5. Plate Number Recognition

In the next step, segmented characters are recognized via OCR. Given the chosen license plate and the coordinates that indicate where the characters are, we begin the OCR process by 2D correlation. We correlate each character with either the alphabet or the numeral templates then choose the value of each character based on the result of the correlation. The first row of standard Bengali License Plates are words containing regional information; therefore, we correlate each one of them with the word templates.

Figure (4.1): Number Recognition 1

The second row contains 7 characters including the hyphen. The latter seven characters are correlated with the 10 numeral templates.

Figure (4.2): Number Recognition 2

The result OCR is chosen based on the maximum values of the correlation for each character. If there is a possibility that a letter or number can be confused with more than one character, the characters are listed in the output in the order of decreasing likelihood. This likelihood is based on the correlation values of the character with the various templates, where high correlation denotes a good possibility. However, this scheme only occurs if the given letter does not have a very high correlation value (does not land above a nominal threshold)[37].

3.6. Data Manipulation and Computation

The license plate number found in the previous step is then stored in the database. We chose MySQL as our preferred database. The license number is stored along with the timestamp and active status. When the license number is detected again, it sets the active status to "no" and calculate start time and end time. In this way the electronic toll is generated. The work flow is shown in the figure below,

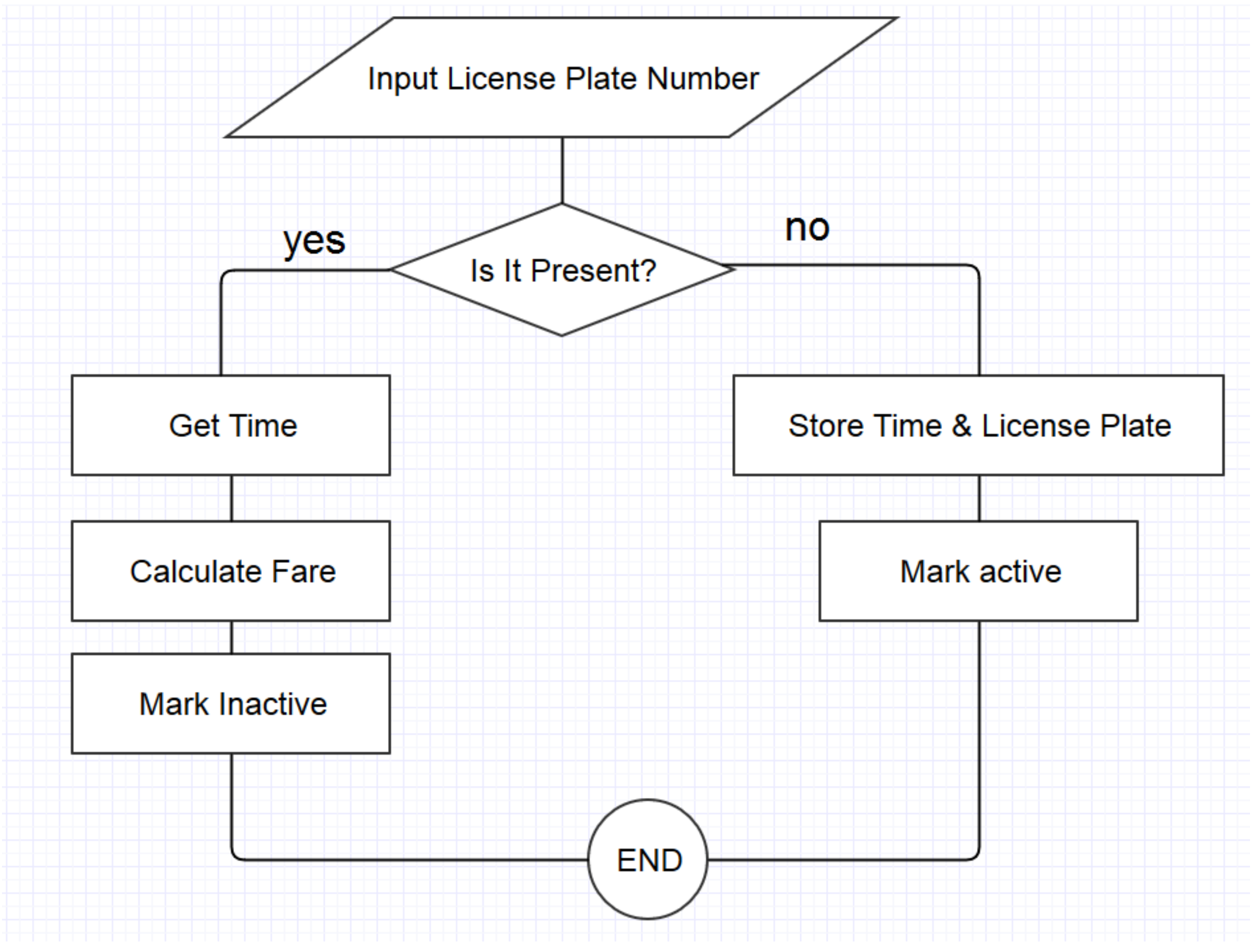


Figure (5): Database Design

4. Experiments and Result Analysis

The system was implemented in MATLAB 2014a and it was tested against 20 different license plates.

4.1. Image Acquisition

The quality of the video feed directly affects the result of the system. 20 different plates were tested with a low quality 5 Megapixel Smartphone Camera and Sony 10.1 MP Cybershot Camera. There was a significant difference in the result.

Table 1

Results of Image Acquisition

Camera Type	**Number of Images**	**Success %**
5 MP Smartphone	20	50
Sony 10.1 MP Cybershot	20	100

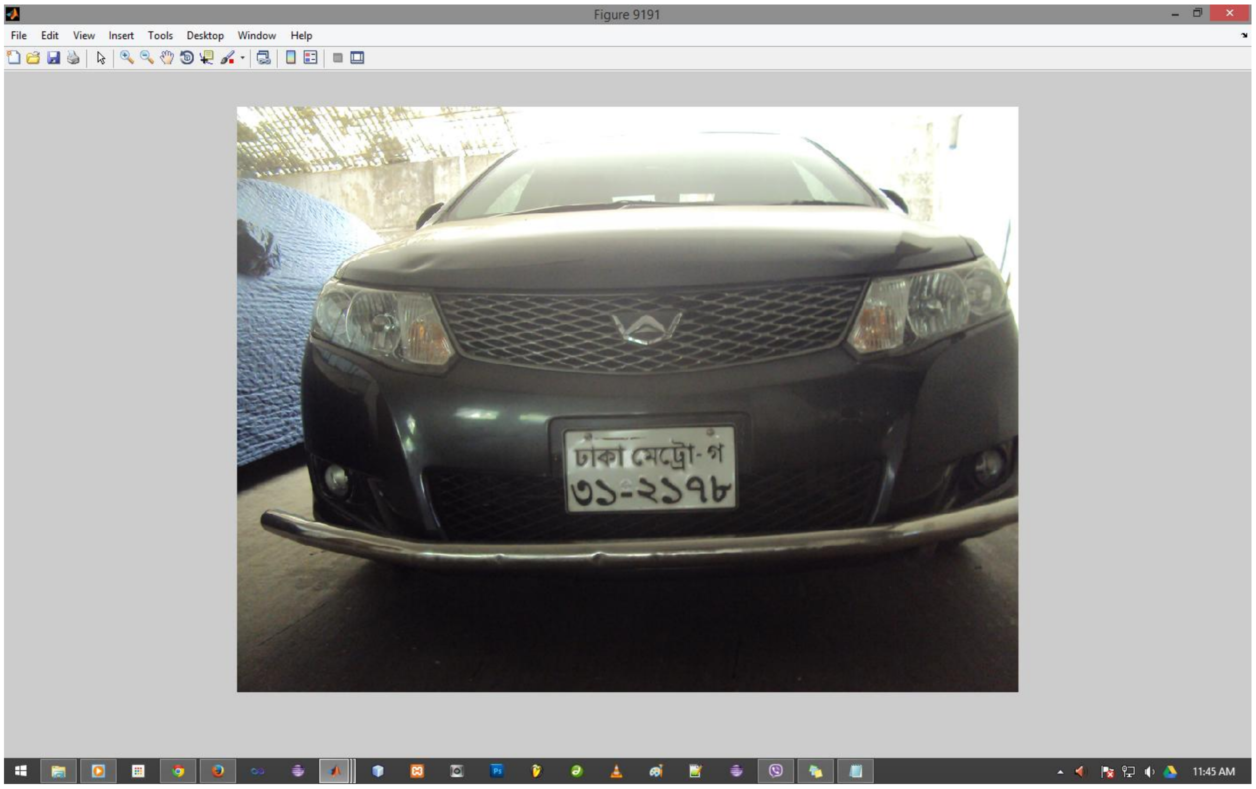


Figure (6.1): Analysis of Image Acquisition

4.2. Preprocessing

4.2.1. Resizing the Image

The images were then tested with original resolution and resized version of the image. The resized image showed higher accuracy because the images were clearer and the edges were more visible. The resized image performed faster than the original picture because of the low resolution.

Table 2

Results of Resizing

Resolution	**Number of Images**	**Success %**	**Performance**
Original (3648 x 273	20	80	0.29s
Resized (600 x 800)	20	100	0.17s

4.2.2 Convert Image to Grayscale

In the next step we convert the image to grayscale from RGB in order to optimize our program. Grayscale is a single channel image while RGB has three channels. For license plates, we need black and white colors so RGB channel is redundant for this system.

Table 3

Results of Grayscale Conversion

Color Channel	**Number of Images**	**Performance**
RGB	20	0.31s
Grayscale	20	0.17s

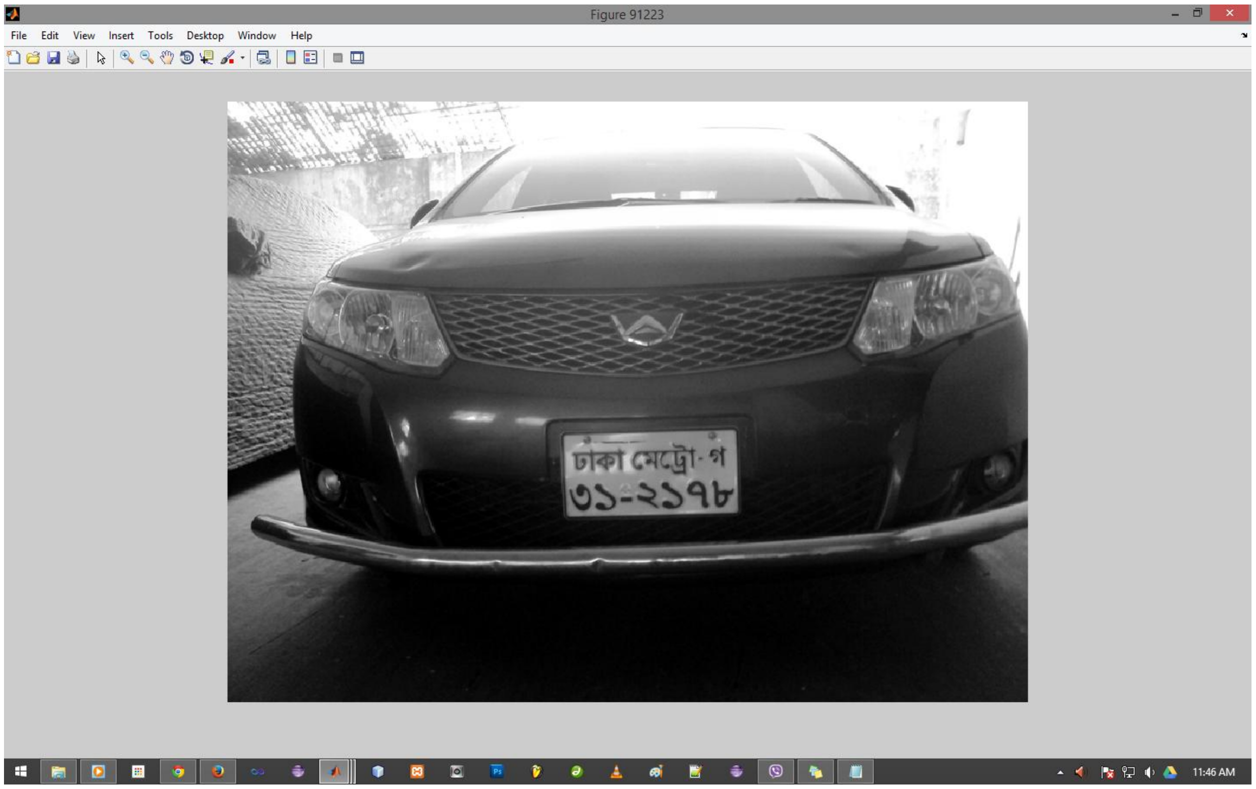


Figure (6.2): Conversion to Grayscale

4.3. License Plate Localization

4.3.1. Edge Detection

We tested the system with different types of edge detection method, for example, sobel, canny etc. Canny edge detection provided us with the most number of connected components which is effective for this system because the accuracy is higher with higher number of connected components.

ঢাকা মেট্রো-গ
৩১-২১৭৮

ঢাকা মেট্রো-গ
৩১-২১৭৮

Figure (6.3): Different Edge Detections

Table 4

Results of Edge Detections

Edge Detection Method	Number of Images	Success %	Avg. Connected Components
Sobel	20	90	873
Prewitt	20	90	897
Canny	20	100	1054

4.3.2. Connected Component Labeling

After detecting the edges, the connected components are distinguished by connected component labeling. This method was compared with window filtering method and connected component labeling showed higher accuracy.

Table 5

Results of Blob Detection

Algorithms	**Number of Images**	**Success %**
Window filtering	20	95
Connected Component Labeling	20	100

4.3.4. Hough Transformation

We go through every connected component and try to find a rectangular object which might be the license plate.

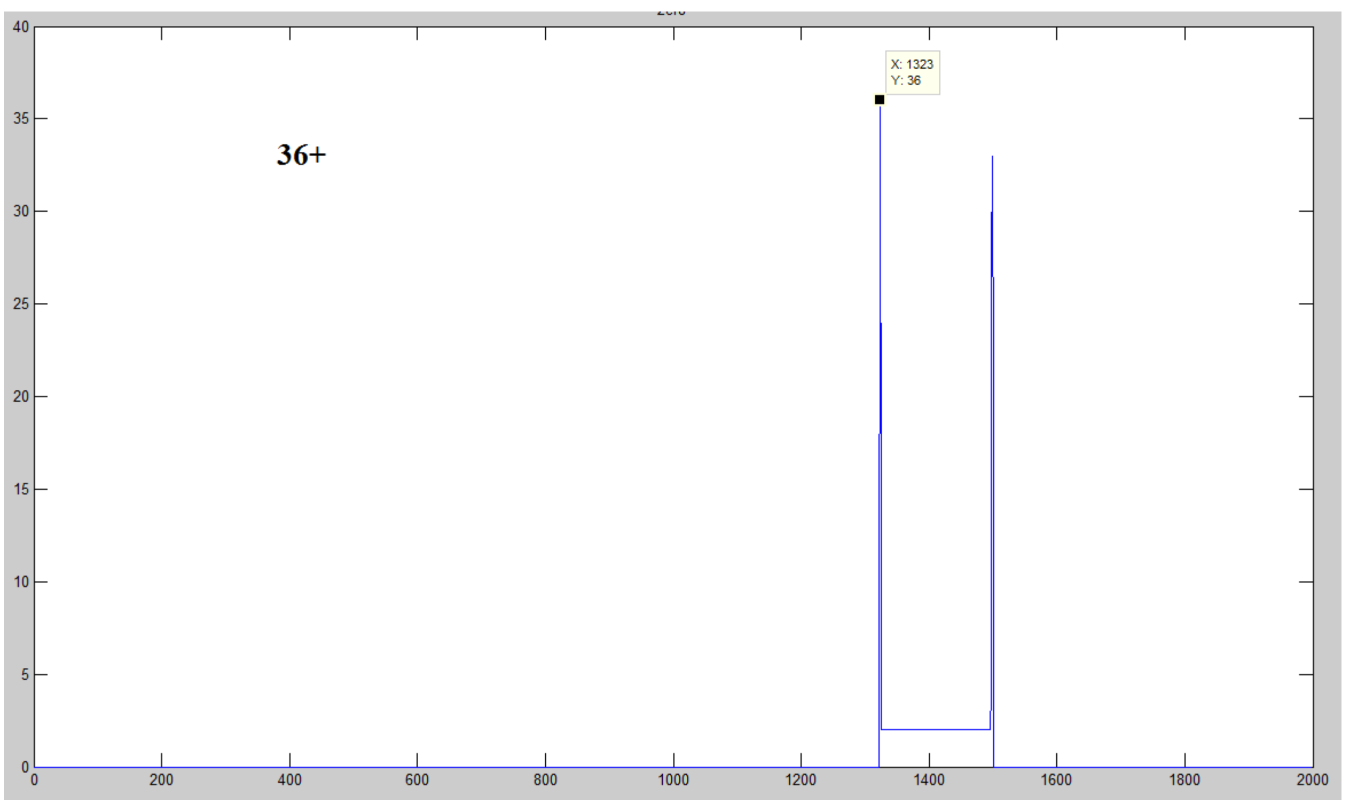


Figure (7.1): Analysis Hough Transformation 1

So, to analyze it, here the graph is showing for the angle of ninety degree. The x and y's values are following by 1323 and 36 indicating at the rho value of 1323 there found a peak value of 36.

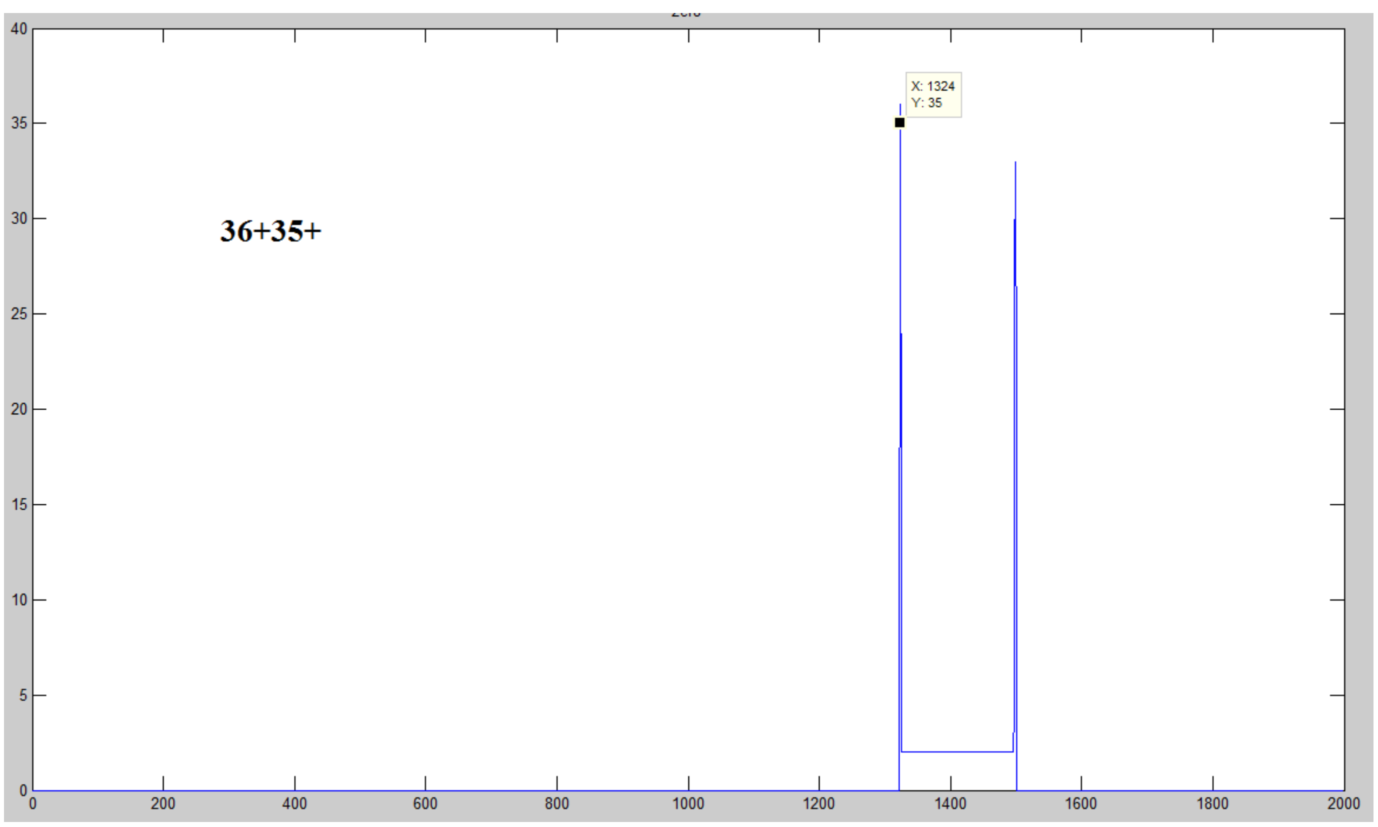


Figure (7.2): Analysis Hough Transformation 2

Here, at the value of rho 1324 the peak value is 35.

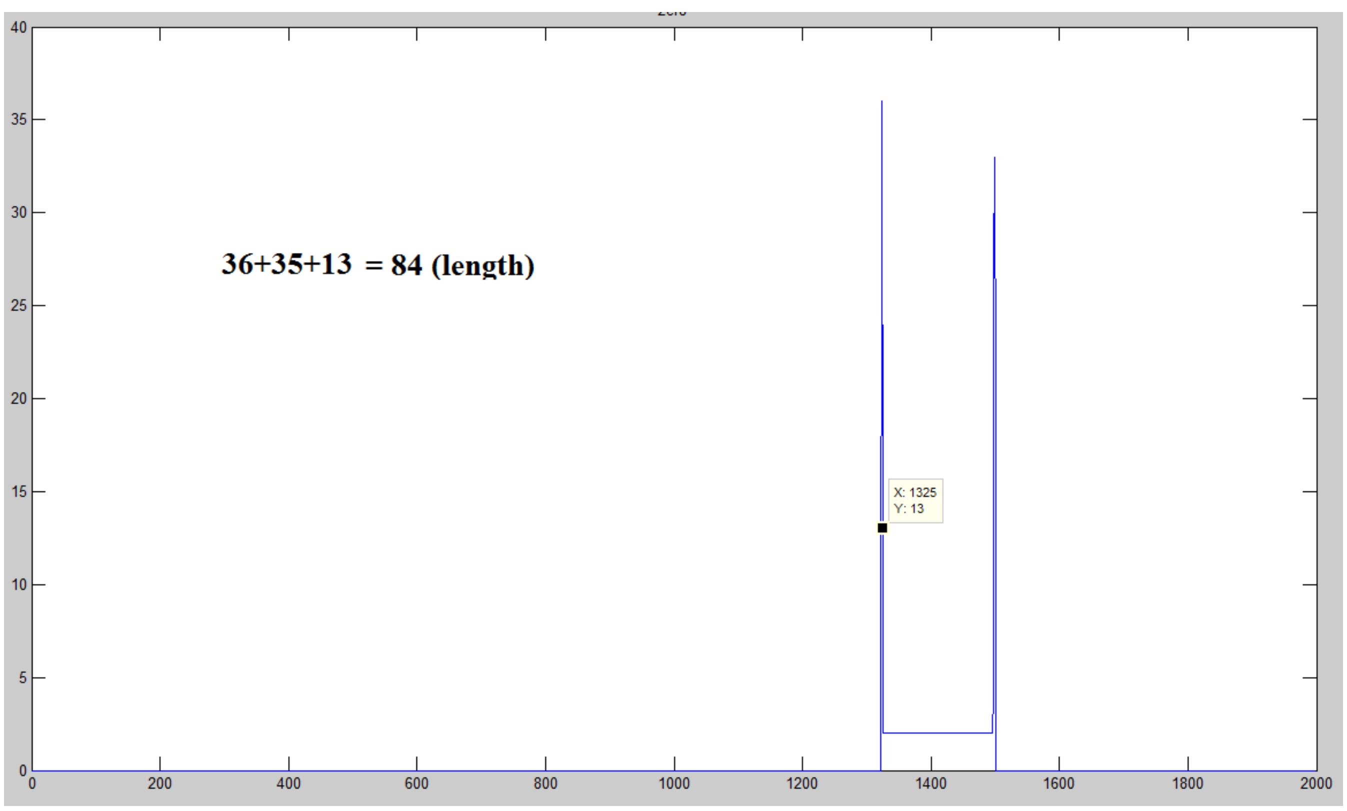


Figure (7.3): Analysis Hough Transformation 3

Here, at the value of rho 1325 the peak value is 13. Adding all up we find the length 84 (unit pixel) for the first line.

Then again if we run the algorithm we find out :

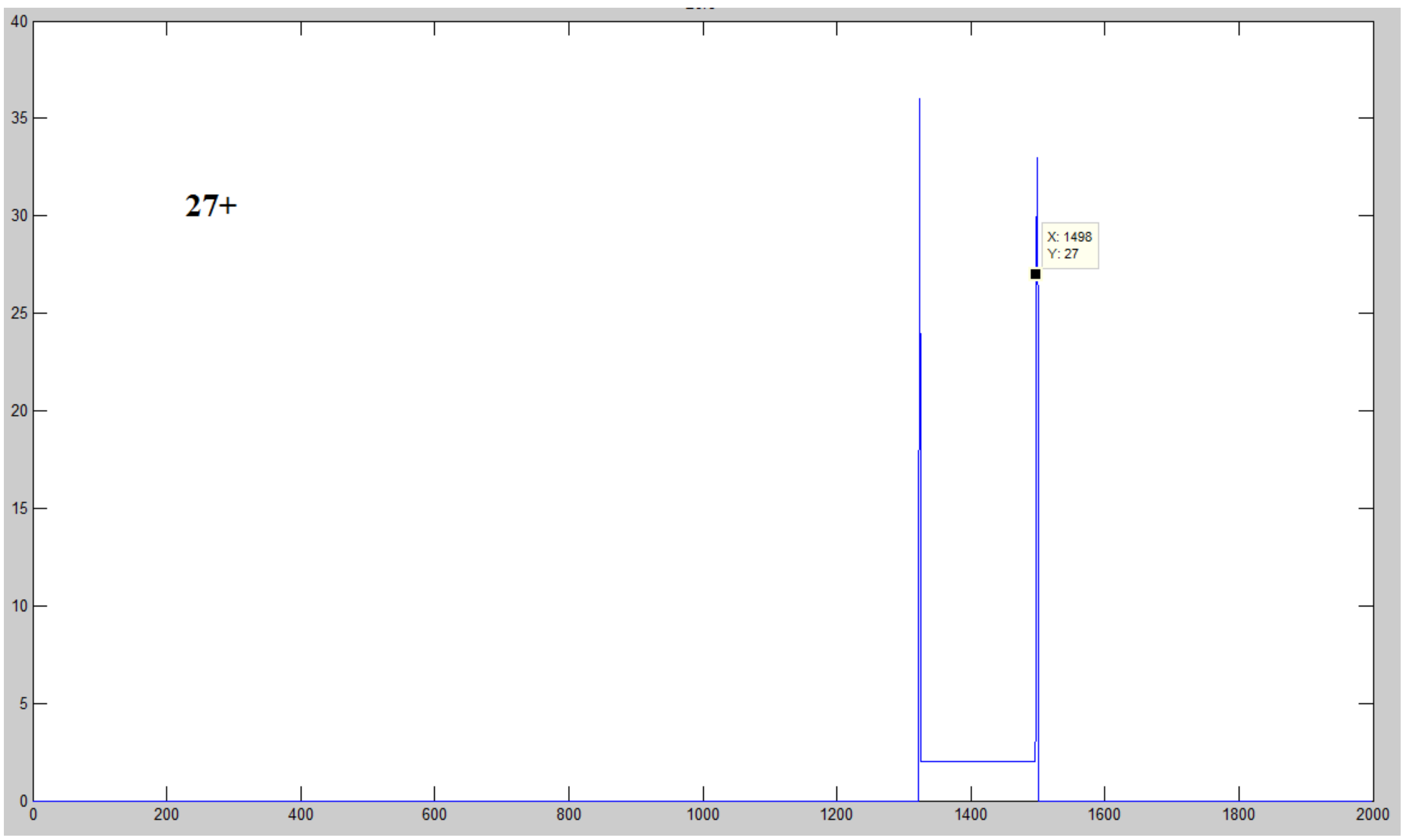


Figure (7.4): Analysis Hough Transformation 4

Here, at the value of rho 1408 the peak value is 27.

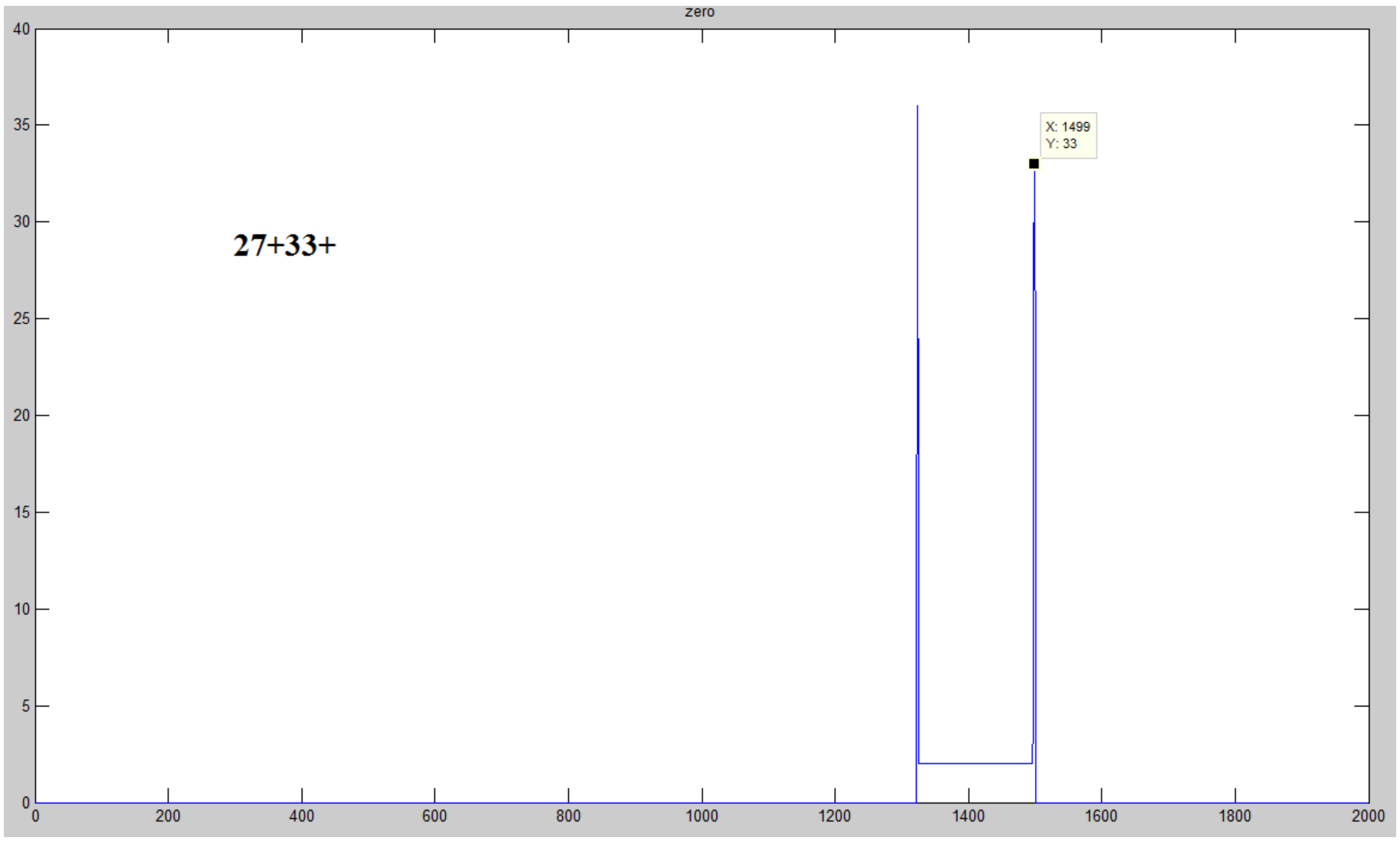


Figure (7.5): Analysis Hough Transformation 5

Here, at the value of rho 1499 the peak value is 33.

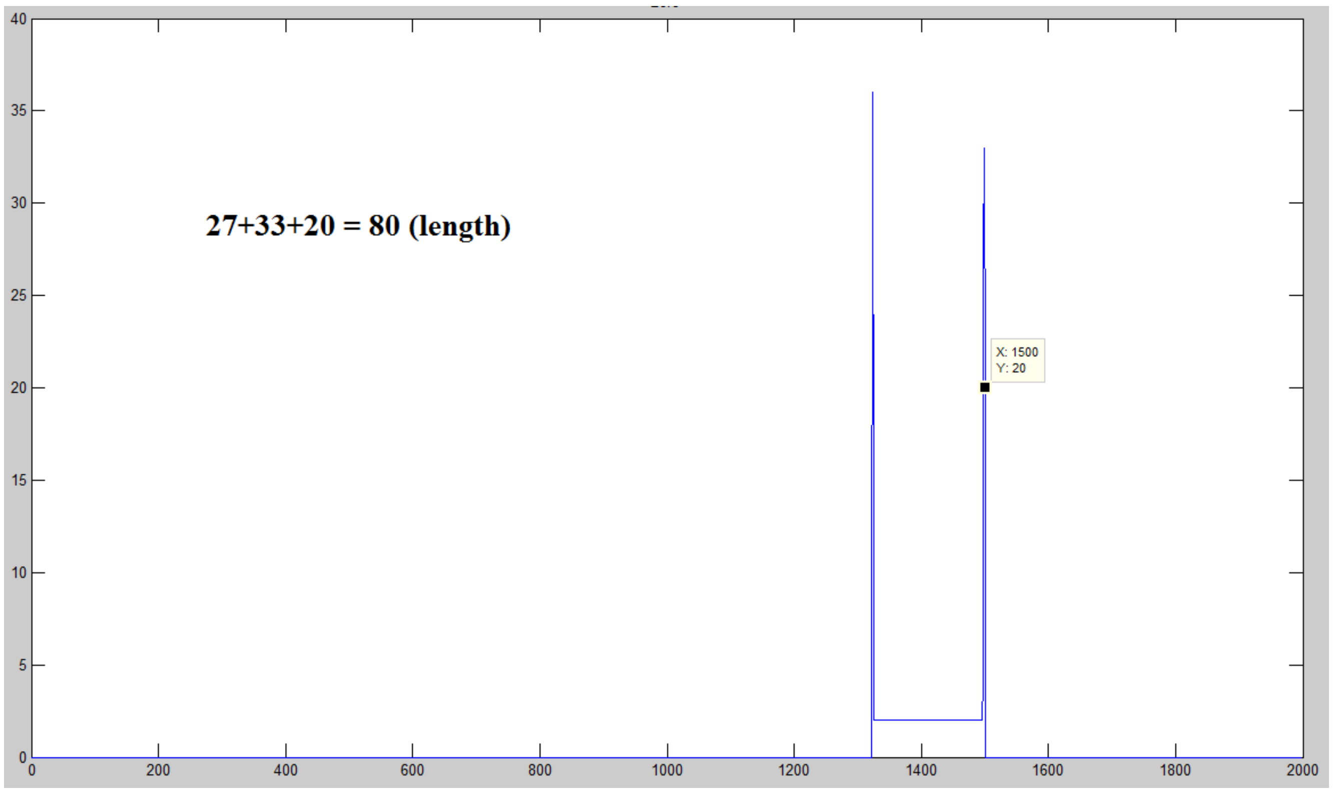
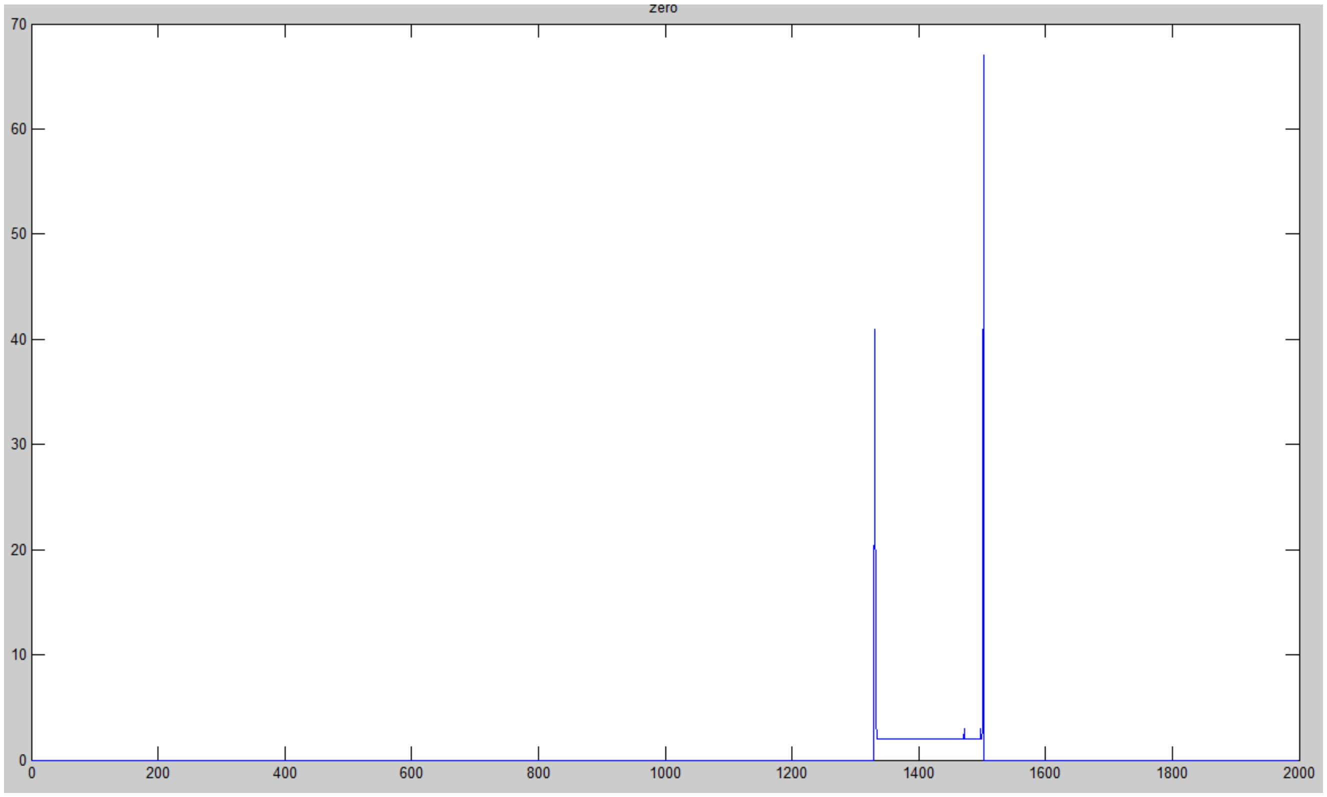


Figure (7.6): Analysis Hough Transformation 6

Here, at the value of rho 1560 the peak value is 20. Adding all up, we find the length of 80 (unit pixel) for the opposite line of the previous one. So, it turns up in the first case we found 84 and in the second case we found 80 of length. A slight difference can be considered as we will mention it on the thresholds. In this way the algorithm finds the length of a rectangle for ninety degree.

So at a glance:



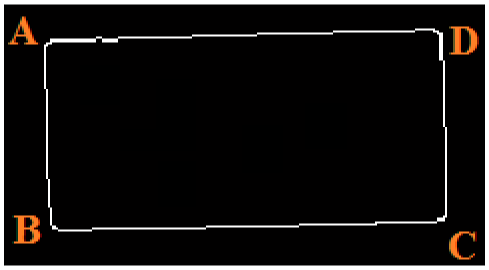
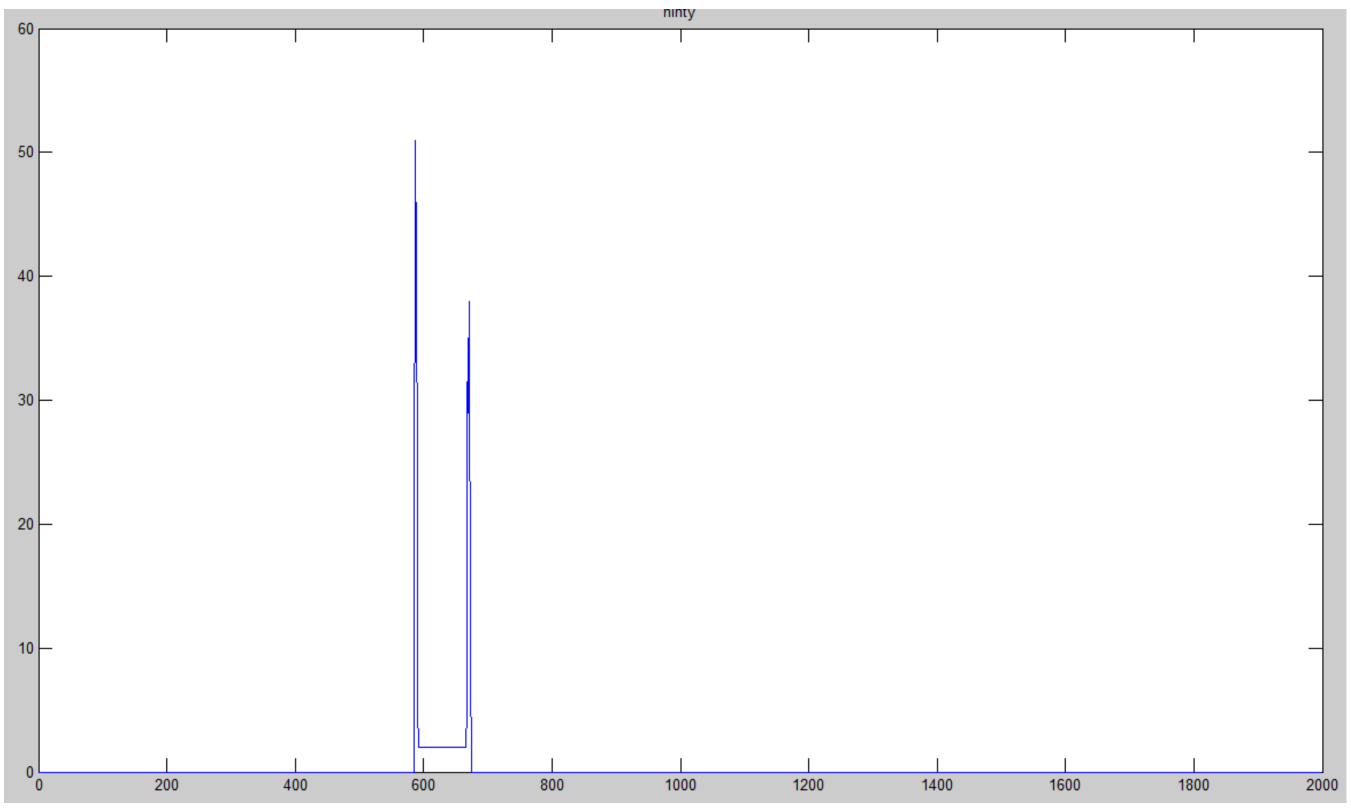


Figure (7.7): Figure (7): Analysis Hough Transformation 7

The graph above is for the zero degree where the x's (value of rho) value varies from 1329 to 1503 indicating the the line AB and CD from the second figure.



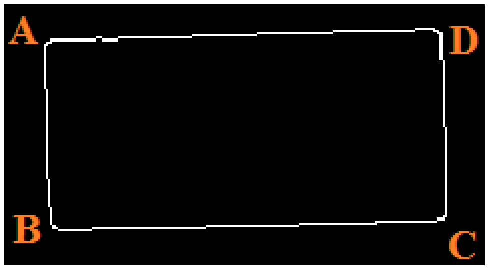


Figure (7.8): Analysis Hough Transformation 8

The graph above is for the zero degree where the x's (value of rho) value varies from 585 to 674 indicating the line AD and BC from the second figure.

After getting all the length of the rectangular license plate, we are ready for the thresholds checking part.

At this point, we have found the license plate completely. It can be something of like below:

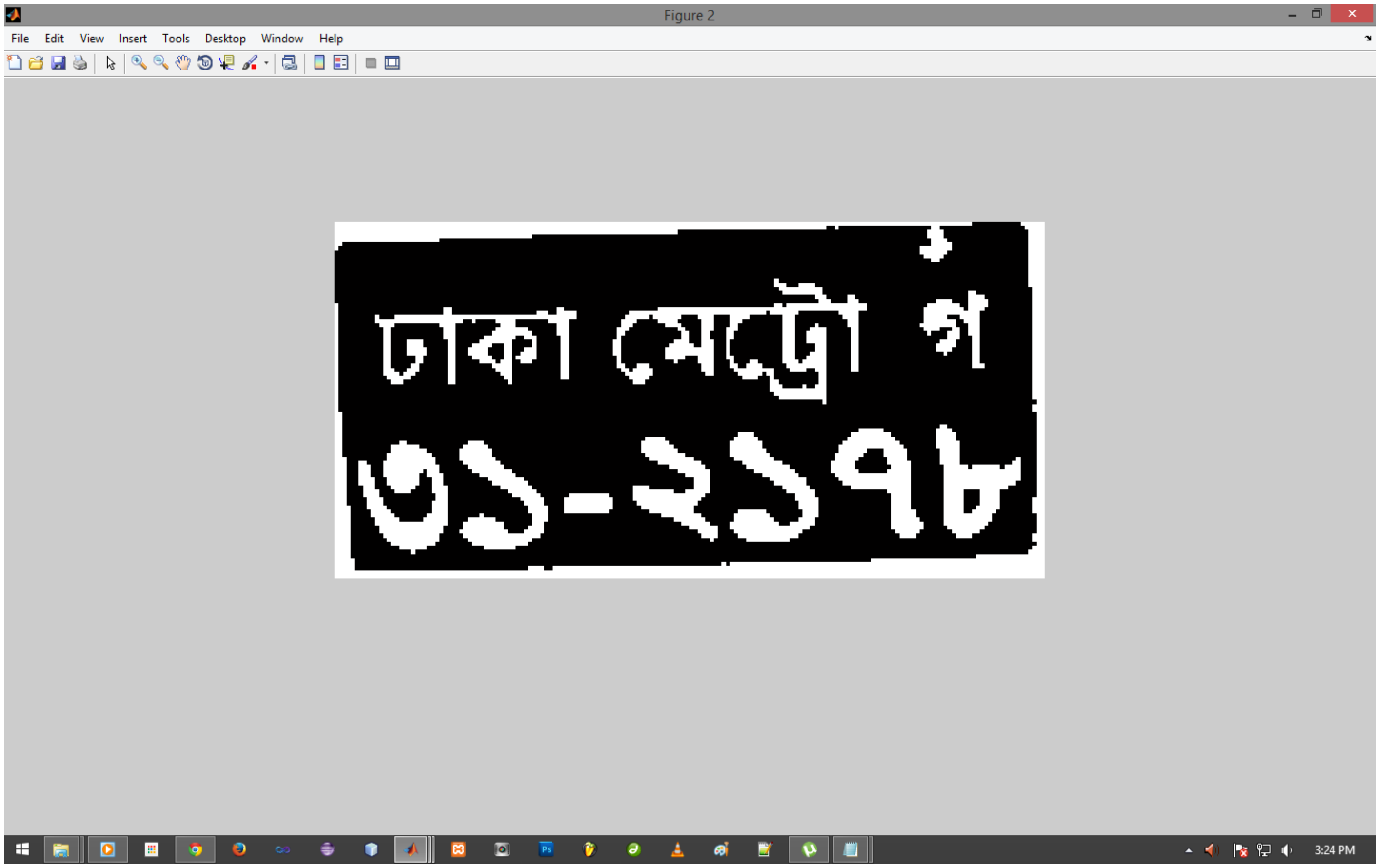


Figure (7.9): License Plate

The modified hough transformation works better than the stock hough transformation of MATLAB.

Table 6

Results of Hough Transformation

Hough Transformation	**Number of Images**	**Success %**
Stock	20	40
Modified	20	100

4.4. Character Segmentation

We chose the horizontal and vertical projection algorithm for character segmentation instead of connected components because the Bengali letters are tricky to handle and connected component showed lower accuracy compared to horizontal projection. The segmented characters and words look like the figure below,

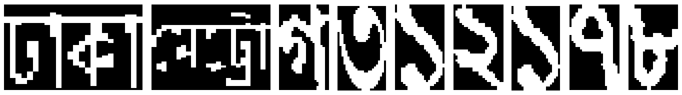


Figure (8): Character Segmentation

Table 7

Results of Character Segmentation

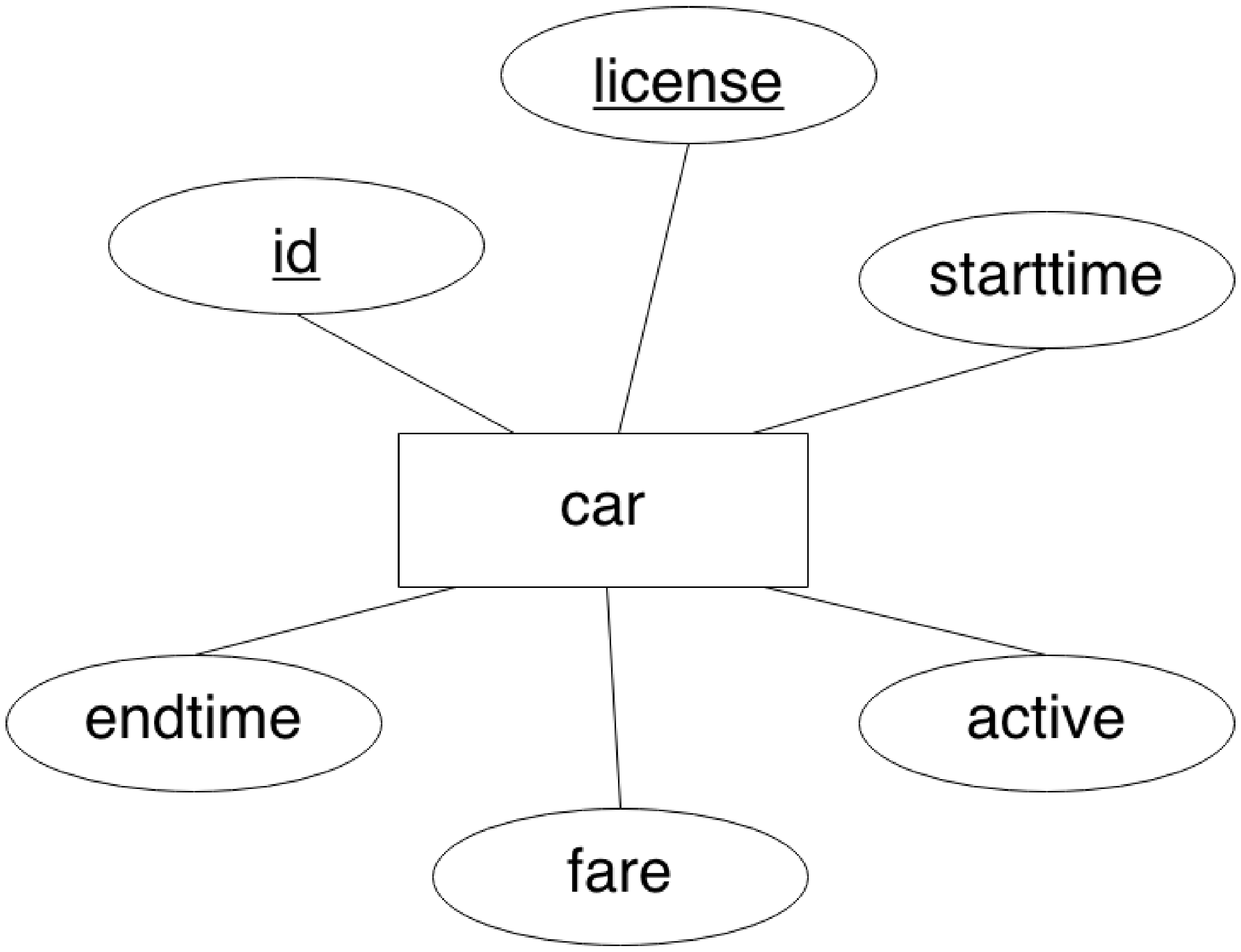
Character Segmentation	**Number of Images**	**Success %**
Connected Components	20	90
Horizontal Projection	20	100

4.5. Plate Number Recognition

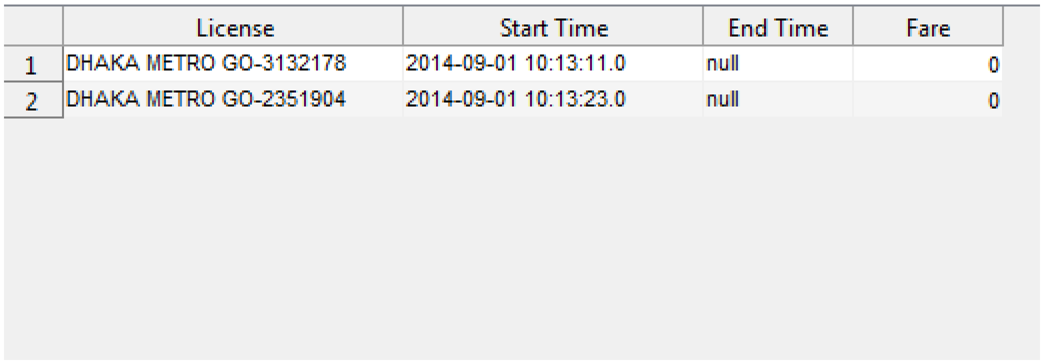
We trained the Tesseract OCR to work with our system, the newly issued licensed plates have same set of fonts, so we used the fonts we received from those license plates as templates. We take the segmented characters and match against the templates to determine which character it is closest to. The accuracy with this method is almost 100%.

4.6. Data Manipulation and Computation

We tried Mysql, Mssql and Access DB for database management system and mysql performed slightly better than mssql and a lot better than Access DB. For computations, regular algebraic functions were used. The ER diagram of the database is shown in the figure below,



This is the database what looks like after the compilation:



	License	Start Time	End Time	Fare
1	DHAKA METRO GO-3132178	2014-09-01 10:13:11.0	null	0
2	DHAKA METRO GO-2351904	2014-09-01 10:13:23.0	null	0

The accuracy from the final product was as much as 100% which is better than the plate detection methods mentioned in reference [38],[19],[20].

5. Conclusion & Future Work

The algorithm used in this system can localize the Bengali license plate using binarization, canny edge detection, connected component labelling, hough transformation. It can also segment the characters and recognize it with OCR. It stores the license and time information and generate electronic toll.

Due to time constraints and lack of experience in image processing, we were unable to make this system as efficient as it could be. There are a numerous improvements that can be made to the license plate.

- Improve and optimize frame cutting from the video feed.
- Improve accuracy when the headlight is turned on or there are disturbing elements in the background.
- Improve accuracy for skewed images.

References

[1] SherrZheng Wang, HsMian Lee, “Detection and Recognition of License Plate Characters with Different Appearences”, in proc. Conf. Intelligent Transportation Systems, vol. 2, pp. 979-984, 2003.

[2] M. M. Rashid, A. Musa, M. Ataur Rahman, and N. Farahana, A. Farhana, "Automatic Parking Management System and Parking Fee Collection Based on Number Plate Recognition", International Journal of Machine Learning and Computing, Vol. 2, No. 2, April 2012

[3] Younghyun Lee1, Taeyup Song, Bonhwa Ku1, Seoungseon Jeon, David K. Han , Hanseok Ko “ License Plate Detection using Local Structure Patterns” in proc. Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance pp 574-579, 2010.

[4] Danian Zheng, Yannan Zhao, Jiaxin Wang, “An efficient method of license plate location, Pattern Recognition Letters,” ISSN 0167-8655, 10.1016/j.patrec.2005.04.014. Vol. 26/15, pp 2431-2438, Nov. 2005,

[5] Preemon Rattanathammawat, Thanarat H. Chalidabhongse, “A Car Plate Detector using Edge Information”, in proc. International Symposium on Communications and Information Technologies, ISCIT '06., pp.1039-1043, October 18-September 20, 2006.

[6] Xifan Shi, Weizhong Zhao, and Yonghang Shen, “Automatic License Plate Recognition System Based on Color ImageProcessing”, Lecture Notes on Computer Science, Springer-Verlag, Vol. 3483, pp. 307-314, 2005.

[7] E.R. Lee, P.K. Kim, H.J. Kim, “Automatic recognition of a car license plate using color image processing”, in proc. IEEE International Conference on Image Processing, pp. 301–305, Austin, Texas, 1994.

[8] X. He, H. Zhang, W. Jia, Q. Wu, T. Hintz, ‘Combining Global and Local Features for Detection of License Plates in a Video’, in proc. of Image and Vision Computing New Zealand 2007, pp. 288–293, Hamilton, New Zealand, December 2007

[9] Otsu, N., A threshold selection method from gray-level histogram, in journal on IEEE Transac

[10] Qadri, M. T. and Asif, M. Automatic Number Plate Recognition System For Vehicle Identification Using Optical Character Recognition; 2009 in proc. International Conference on Education Technology and Computer. IEEE Xplore; pp.335-338. 2009.

[11] Saleh, Marwan D.; Mellah, H.; Mueen, Ahmed; Salih, N. D.; , "An efficient method for vehicle license plate xtraction," International Symposium on Information Technology, 2008. ITSim 2008, vol.2, no., pp.1-5, 26-28 Aug. 2008.

[12] B. Fröba, and A. Ernst, "Face Detection with the Modified Census Transform", in proc. Sixth International Conference on Automatic Face and Gesture Recognition (FGR'04), pp. 91-96, Seoul, Korea, May 17-19, 2004.

[13] Babu, C.N.K.; Nallaperumal, K., "A license plate localization using morphology and recognition," India Conference, 2008. INDICON 2008. Annual IEEE, vol.1, no., pp.34-39, 11-13 Dec. 2008.

[14] Wei-Hsun Lee, Shian-Shyong Tseng, Ching-Hung Wang, "Design and implementation of electronic toll collection system based on vehicle positioning system techniques" in Journal on Computer Communications, Elsevier Science Publishers B. V. Amsterdam, The Netherlands, Vol. 31/12, pp 2925–2933 July 2008.

[15] Faradji, F.; Rezaie, A.H.; Ziaratban, M.;, "A Morphological-Based License Plate Location", in proc. IEEE International Conference on ImageProcessing, 2007, Vol. 1, pp:1 - 57 - 1-60, 2007.

[16] IAITO INFOTECH PVT LTD, "RFID based Vehicle Tracking & Parking Lot Management System"

[17] Automated Number Plate Recognition –
"http://en.wikipedia.org/wiki/Automatic_number_plate_recognition#Development_history"

[18] Jason Grant, "Automatic License Plate Recognition"

[19] Mahmudul Hasan, "Real Time Detection and Recognition of License Plate in Bengali"

[20] Mashuk M. S., Majid M. A., Basher N. and Rahman T. R., "Automatic Detection of Bangla Characters in Bangladeshi Car Registration Plates," Second International Conference on Computational Intelligence, Modelling and Simulation, pp. 166-171, 2010

[21] Optical Character Recognition Software - "http://en.wikipedia.org/wiki/Optical_character_recognition"

[22] First Bengali OCR Released - "http://bdnews24.com/technology/2014/08/08/first-bengali-ocr-launched"

[23] CRBLP Bangla OCR - "http://crblpocr.blogspot.com"

[24] OmriconLAB - "http://omriconlab.com"

[25] MATLAB rgb2gray - "http://www.mathworks.com/help/images/ref/rgb2gray.html"

[26] C.Rajkumar, S.K.Mahendran - "Vehicle Detection and Tracking System from CCTV Captured Image for Night Vision", IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 6, August 2014.

[27] Edge Detection - "http://en.wikipedia.org/wiki/Edge_detection"

[28] Canny Edge Detector - "http://en.wikipedia.org/wiki/Canny_edge_detector"

[29] R. Fisher, S. Perkins, A. Walker and E. Wolfart (2003). "Connected Component Labeling".

[30] “Pixel connectivity-wikipedia” - http://en.wikipedia.org/wiki/Pixel_connectivity

[31] Haralick, Robert M., and Linda G. Shapiro, Computer and Robot Vision, Volume I, Addison-Wesley, 1992, pp. 28-48.

[32] “Minimum bounding box” - http://en.wikipedia.org/wiki/Minimum_bounding_box

[33] Hough Transformation - "http://en.wikipedia.org/wiki/Hough_transform"

[34] O. Martinsky, “Algorithmic and Mathematical Principles of Automatic Number Plate Recognition System”, B. Sc. Thesis, BRNO University of Technology, 2007.

[35] Satadal Saha, Subhadip Basu, Mita Nasipuri, Dipak Kumar Basu “An Offline Technique for Localization of License Plates for Indian Commercial Vehicles”

[36] Wisam Al Faqheri and Syamsiah Mashohor, "A Real-Time Malaysian Automatic License Plate Recognition (M-ALPR) using Hybrid Fuzzy", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.2, February 2009.

[37] Anagnostopoulos C. E., Anagnostopoulos I. E., Loumos V., and Kayafas E., A License Plate-Recognition Algorithm for Intelligent Transportation System Applications, IEEE Transactions On Intelligent Transportation Systems, vol. 7, no. 3, pp. 377-382, 2006.

[38] Duan T. D., Hong T. L., Phuoc T. V., and Hoang N. V., ”Building an Automatic Vehicle License-Plate Recognition System,”Intl. Conf. in Computer Science RIVF05, Can Tho, Vietnam, 2005.

[39] Mashuk M. S., Majid M. A., Basher N. and Rahman T. R., ”Automatic Detection of Bangla Characters in Bangladeshi Car Registration Plates,” Second International Conference on Computational Intelligence, Modelling and Simulation, pp. 166-171, 2010.