# Shahjalal University of Science and Technology
## Department of Computing Science and Engineering


# Automated Bangla License Plate Recognition

Md. Al-amin Nowshad
Reg. No.: 2012331059
$4^{th}$ year, $1^{st}$ Semester

Sudipto Chandra Das
Reg. No.: 2012331019
$4^{th}$ year, $1^{st}$ Semester

Department of Computer Science and Engineering

**Supervisor**
Md. Jahirul Islam
Professor
Department of Computer Science and Engineering

3rd April, 2017

# Automated Bangla License Plate Recognition


A Thesis submitted to the

Department of Computing Science and Engineering

Shahjalal University of Science and Technology

Sylhet - 3114, Bangladesh

in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering

By

Md. Al-amin Nowshad

Reg. No.: 2012331059

$4^{th}$ year, $1^{st}$ Semester

Sudipto Chandra Das

Reg. No.: 2012331019

$4^{th}$ year, $1^{st}$ Semester

Department of Computer Science and Engineering

**Supervisor**

Md. Jahirul Islam

Professor

Department of Computer Science and Engineering

3rd April, 2017

# Recommendation Letter from Thesis Supervisor

The thesis
entitled "Automated Bangla License Plate Recognition"
submitted by the students

1. Md. Al-amin Nowshad
2. Sudipto Chandra Das

is a record of research work carried out under my supervision and I, hereby, approve that the report be submitted in partial fulfillment of the requirements for the award of their Bachelor Degrees.

Signature of the Supervisor:
Name of the Supervisor: Md. Jahirul Islam
Date: 3rd April, 2017

# Certificate of Acceptance of the Thesis

The thesis

entitled "Automated Bangla License Plate Recognition"

submitted by the students

1. Md. Al-amin Nowshad
2. Sudipto Chandra Das

on 3rd April, 2017is, hereby, accepted as the partial fulfillment of the requirements for the award of their Bachelor Degrees.

| | | |
|---|---|---|
| Head of the Dept. | Chairman, Exam Committee. | Supervisor |

# Abstract

*This is primary abstract. It may change in future.*

Automated license plate recognition is important in many contexts like- security and law enforcement, monitoring vehicles, automated parking control. To enable these automated services in Bangladesh we are proposing an efficient method in this paper. The task of recognizing vehicle number plate consists of three stage: plate detection, character segmentation and character recognition. We reviewed several methods for each stage, and choose the best one. The main objective of this research is to gain high accuracy in an efficient manner, keeping into consideration the facts of lighting conditions, vehicle motion, noisy plates, segmented words. Our primary target is to recognize the standard bangla license plates. But we also tested our methods on non-standard license plates. For standard license plate we have gained **XX%** accuracy in plate detection, **XX%** in character segmentation and **XX%** in recognition, resulting in **XX%** accuracy overall.

**Keywords:** Automated License Plate Recognition (ALPR), Bangla Number Plate, Image Enhancement, Number Plate Detection, Bangla Character Segmentation, Bangla Character Recognition.

# Acknowledgements

We would like to thank the Department of Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet 3114, Bangladesh, for supporting this research. We are very thankful to our honorable teacher Prof. Md. Jahirul Islam for his outstanding directions and help.

We are also grateful to anonymous authors of previous works.

# Dedication

We would like to dedicate our research to our parents. We are also grateful to anonymous authors of previous works for their co-operation and support.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

In the field of Image processing and computer vision license plate recognition is an well established title. This thesis tries to implement this established title for license plates written in Bengali language with the purpose of making the previous system well organized and more efficient.

Automated license plate recognition systems are available for commercial use with great cost for other languages used in mainly fields like - traffic history check, security maintenance, parking, etc.

## 1.1 Objectives

The main objective of this thesis is to find out an efficient way to implement the ALPR system for Bengali digital license plates. The main objective is divided into sub objectives like - turning the system into a full working application which should be open sourced and organized for further use.

## 1.2 Challenges

For fulfilling objectives there are few challenges to conquer. These challenges are -

1. reviewing and finding lacking of previous systems
2. implement multiple systems to get a better review and future planning

3. gather license plate image dataset
4. preprocessing dataset properly and handling different cases
5. implementing two different main sections - license place extraction and character recognition properly for Bengali language.
6. comparing and performance enhancement of the developed system.

## 1.3 Motivation

As there is no established ALPR system in Bangladesh although few thesis related to this field already been done, it is time to make something properly usable and complete. As the main thought of making a real world effect, it is needed to find and implement a way to make a working Bengali ALPR system.

## 1.4 Approaches

Primarily the approach of this thesis has to be able to make a system working with the limited resources with low cost. Industrial ALPR systems are far more expensive and takes a huge amount of time, money, data which is not affordable in the scope if this thesis. So, ordinary camera images are used as dataset with minimal resolution.

The license plate extraction part is done trying few traditional techniques and also with some out of the box techniques which will be described in the later chapters. And further iterations of tuning the current system will be made in the next section of time frame for this thesis work.

Optical character recognition is the second part of this research work. This is itself a big field which includes past researches for Bengali language. Most of those researches are not of good quality to implement directly in this thesis. But, the datasets can be re-used to for training purposes. So, two approaches are under consideration- using Google's Tesseract OCR and using previous works for Bangla OCR. Both will be tried in the next part of this thesis work.

## 1.5 Thesis Structure

The structure of this thesis report is mainly divided into three parts - introductory discussion including history and other background knowledges related to this field, the technical implementation and review, result analysis and conclusion.

# Chapter 2

# Background Study

## 2.1 Historical View

As far as we know that ANPR system was first invented in 1976. It was done by the Police Scientific Development Branch in the UK. Industrial usage begun in 1979. But ANPR got more known in the 1990's for their cheap implementation from the previous costly systems. The first ANPR dataset was made in early 2000s. This system was used to solve a murder occurred in November 2005, in Bradford, UK, to locate and find the culprits behind the killing of Sharon Beshenivsky.

From that point forward tag acknowledgment has advanced and been utilized by police and different associations for various purposes and has turned out to be less expensive and speedier, yet not sufficiently shoddy for private utilize. In the following areas, a portion of the current techniques in the zone of tag location and acknowledgment will be talked about.

## 2.2 Literature Review

The first task was to find the problem domain to work on. The next task was to make literature reviews for our work plan and also for ensuring that we are not doing something again or worse. We've found few papers related to ALPR for Bengali languages and many more in different languages. Finally, we did select 24 papers to read and review for our thesis work. We've written 7 review papers those seemed close to our topic. From those papers and reviews we've managed

to find and understand our particular domain of work which will be described in the later parts.

## 2.3 Standard Bangla license plate

### 2.3.1 Vehicle types

Vehicles are of different types in Bangladesh. As a result license plates don't stay at the same place for each vehicle. Vehicles considered in this thesis work are - private car, bus, cng auto-rickshaw, truck. Though there were no previous format and standardization of the license plate in the past, currently government has enforced digital license plate for all vehicles which makes it a new case to handle and less error prone than the previous systems.

### 2.3.2 Structure of the plate

The digital Bengali license plate is normally written in two lines.

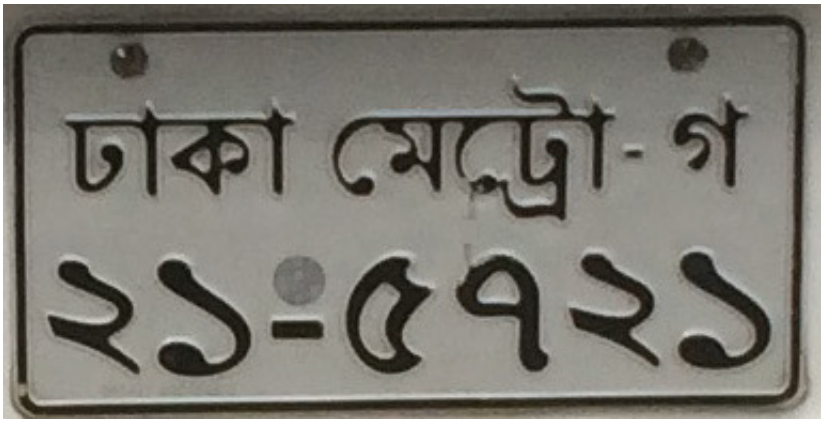



Figure 2.1: Example of a digital license plate in Bangla

As the example given the line on the top written in Bangla and the bottom line has all digits in Bangla.

### 2.3.3 Parts of plate number

The top line got two parts. First part before the '-' denotes the city of the car license and then the second part denotes the series it is in.

The bottom line is the number of the license assigned to the particular vehicle by the licensing rules and regulation of Bangladesh Road Transport Authority.

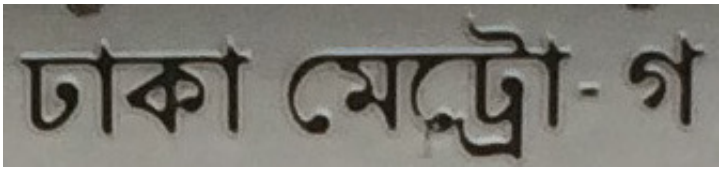



Figure 2.2: upper portion of license plate


Figure 2.3: bottom part of license plate

### 2.3.4 Plate fonts

Digital license plate is offering verified fonts, but in Bangladesh those rules are not strictly followed by all vehicle owners. Also the current dataset doesn't have any universal font. So, the fonts can be different but easily understandable for this thesis work.

## 2.4 Related Works

The rest of the chapter will discuss some of the previous works related to license plate recognition.The whole task is divided into five main steps: Preprocessing, Feature analysis, Region analysis, Character segmentation, Recognition.

These steps are going to be discussed briefly in this chapter.

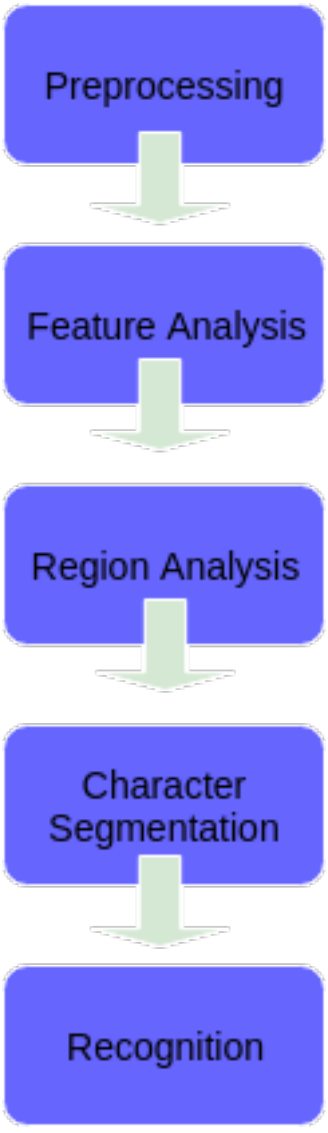



Figure 2.4: Steps of ALPR system

## 2.5 Preprocessing

Preprocessing techniques are used to get a better performance in the next sections. This is one of the key task after image acquisition.It ensures good data input for the main parts of the license plate detection and recognition system.

Most common approach is done in two steps. The image is converted to a gray scale image format to the next steps (Waterhouse-2006). Then, any additional noise is reduced from the image with different types of algorithms in different cases. Most common techniques for reducing noise from input image are the Median filter(Songke and Yixian-2011) and Gaussian filter(Sedighi and Vafadust 2011). Contrast enhancement is an additional technique followed by many researchers(Abolghasemi and Ahmadyfard - 2009) in this pre-processing part.

## 2.6 Feature Analysis

For detecting the license plate, feature analysis must be done after an image is preprocessed properly. Checking the edges for detecting any rectangle shaped area is the most common technique for feature analysis. To detect the plate boundary, some researchers use horizontal lines and others look for vertical lines. Edge detection is commonly done with Canny (Kong, Liu, Lu, and Zhou-2005) and Sobel edge detection (Jiao, Ye, and Huang 2009). As foe some exception some researches do not use edge detection, but try to find high contrast area (Huang, Chang, Chen, and Sandnes-2008). The theory behind this technique is - plates generally have higher contrast area than other parts of the image.

# Chapter 3

# Methodology

This chapter describes our process to detect license plate given an input image, and separation of the individual numbers and character sequence from detected license plate. The input image may come from variety of sources. We have put on a validation method to detect if there is actually a license plate in the input image.

## 3.1 Implementation

We implemented our solution using *Python-3.x* using the *OpenCV-3* and *Tesseract* libraries from Anaconda repository. The *OpenCV* library provides an extensive collection of image processing functions and techniques which proved very helpful in our implementation. *Tesseract* is an open source OCR engine made by Google. Among other libraries, *numpy* has been used the most to manipulating 2D arrays.

For testing and analysis we used a separate implementation. In this implementation, we divided entire tasks into sequence of **stages**. This *stages* act like individual modules. For example, a stage for Gaussian filter will only apply Gaussian filter to a set of input image and save the output to another folder to make it useful as input for later stages. This approach on testing proved to be quite useful and saved a lot of time during our research.

A flow chart will be here containing:
1. Input Image
2. Preprocessing
3. Plate localization
4. Region analysis
5. Plate recognition

Figure 3.1: An overview of the plate detection procedure.

## 3.2 Overview

We separate our process into several stages. Each stages contains several steps in it. An overview of all stags in our process is shown in Figure 3.1. The process will be described in more details in the following sections.

**Input Image** is a 24-bit colored image with red, green and blue channels.

**Preprocessing** stage applies several operations on input image to convert it into an image suitable for feature analysis for later stages.

**Plate detection** analyses content of the image to identify and extract the plate like regions.

**Region analysis** cleans the extracted plate (removing borders, noise etc), and segments the plate into characters. Also removes bad estimations.

**Plate recognition** makes the image ready for *Tesseract*.

## 3.3 Preprocessing

This stage contains a sequence of steps that enhance the plate like regions of an image and output an image that can be used directly for the next stage - plate detection.

### 3.3.1 Rescale

By minimizing the size of an image we can reduce processing time dramatically. But reducing image may result in loss of information. From reviews, we decided to use the dimension: 640×480. We re-scale the input image into this dimension at first step. Figure 3.2 shows a sample image.


Figure 3.2: A sample input image (rescaled)

### 3.3.2 Gray Scale Conversion

From the 24-bit color value of each pixels $(i, j)$ of the input image, we split the $R$, $G$, $B$ components . Then, 8-bit gray value is calculated using Equation 3.1. A sample result is shown in Figure 3.3.

$$gray(i, j) = 0.59 * R(i, j) + 0.30 * G(i, j) + 0.11 * B(i, j) \tag{3.1}$$

### 3.3.3 Vertical Edge Density

The vertical *Sobel operator* (Equation 3.2) is used to calculated the edge density.

$$h = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{3.2}$$

Which we could obtain by using *sobel* function from OpenCV library:


Figure 3.3: A sample gray-scale image

```
sobel = cv2.Sobel(img, cv2.CV_8UC1, 1, 0, ksize=3)
```

Next, we applied a low threshold value to the gradient image. In this case, we used an adaptive threshold technique called *Otsu's Binarization* using an offset value of 85, which we obtained empirically. Result is shown in Figure 3.4

### 3.3.4 Gaussian Filter

In this step, 2D Gaussian filter is applied to the gradient image from previous step. We used a Gaussian kernel of size $60 \times 60$. To calculate the kernel the formula in Equation 3.3 is used. Figure 3.5 shows a plot of this filter.

$$g(i, j) = \frac{\alpha}{2\pi\sigma^2} \cdot e^{\frac{i^2+j^2}{2\sigma^2}} \tag{3.3}$$

Here, the blur coefficient $\alpha$ and the $\sigma$ are set empirically. We used $filter2D$ function to apply convolution of on the gradient image

```
gauss = cv2.filter2D(sobel, cv2.CV_64F, kernel)
```

As a result we get a nicely blurred image (Figure 3.6).

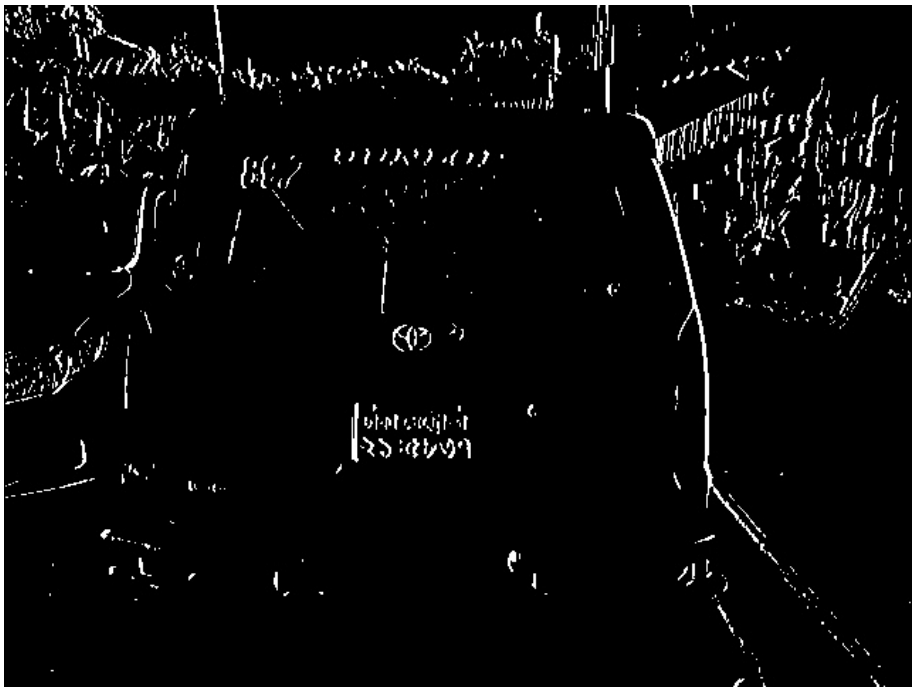


Figure 3.4: After applying Sobel Operator

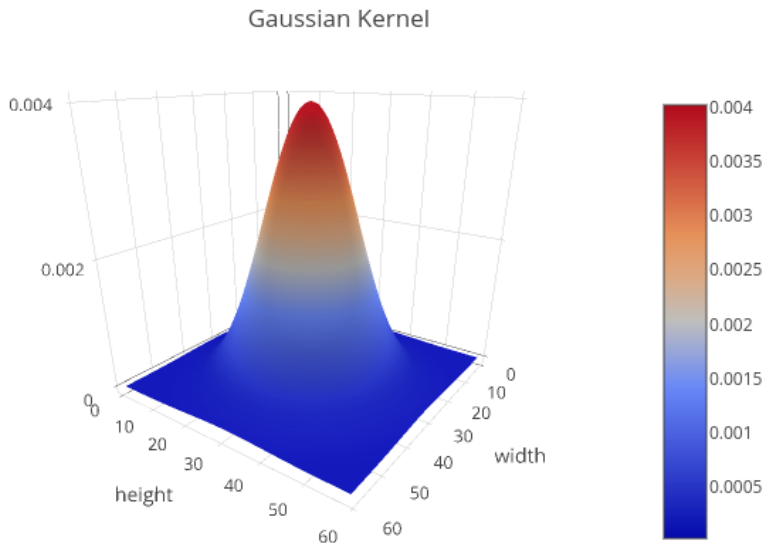



Figure 3.5: The Gaussian filter

Figure 3.6: Applied the Gaussian Blur

### 3.3.5 Image Enhancement

We used the Gaussian image to increase the contrast of the image around plate like regions. The formula used is shown in Equation 3.4.

$$I' = f(\rho W_g)(I - \bar{I}) + \bar{I} \tag{3.4}$$

In the equation 3.4,

$\bar{I}$ = the mean intensity.

$I$ = the intensity of the original image.

$I'$ = the new enhanced image.

$W_g$ = normalized Gaussian image.

$\rho W$ = the standard deviation of Gaussian image, and

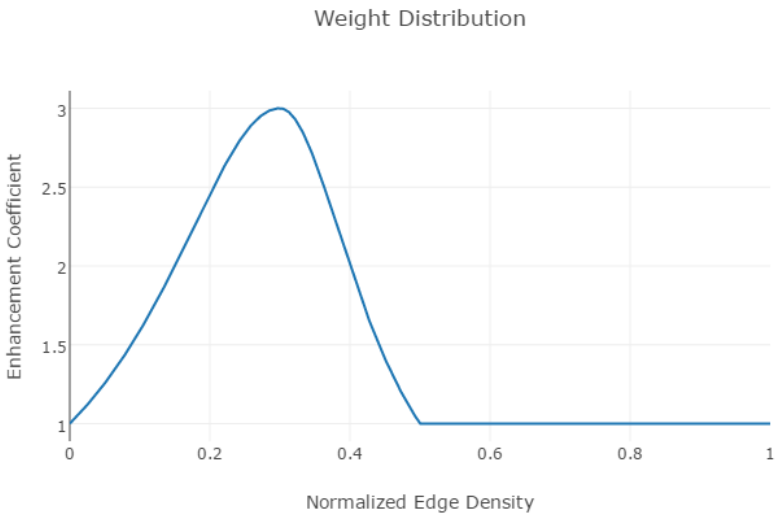



Figure 3.7: Weight function against normalized edge density

$f$ = the weighting function defined in Equation 3.5.

$$f(\rho W_g) = \begin{cases} \dfrac{3}{\dfrac{2}{0.3^2}(\rho W_g - 0.3)^2 + 1}, & \text{if } 0 \le \rho W_g < 0.3 \\ \dfrac{3}{\dfrac{2}{(0.5-0.3)^2}(\rho W_g - 0.3)^2 + 1}, & \text{if } 0.3 \le \rho W_g < 0.5 \\ 1 & \text{if } \rho W_g \ge 0 \end{cases} \tag{3.5}$$

Figure 3.7 shows the map of weight function from 3.5.

To increase the efficiency of the calculation the entire image is divided into $8 \times 8$ windows each having the size of $60 \times 80$. For each window we calculated weight and mean intensity using *Bilinear interpolation*. Also by using *numpy* library for matrix manipulation, we could dramatically decrease processing time.

Figure 3.8 shows the resulting image. The brightness of license plate area has significantly improved.


Figure 3.8: Enhanced Image

### 3.3.6 Matched Filter

To detect the license plate, next step is to highlight plate like regions from the enhanced image. We used a mixture of Gaussian functions to emphasize the constancy of intensity values within plate-like regions along horizontal direction. Equation 3.6 shows the mathematical model of this function. This function can properly model low edge densities above, below and at the middle of the car plate.

$$h(x,y) = \begin{cases} A.exp\left(\dfrac{-\left(x-\dfrac{m}{6}\right)^2}{0.2\sigma_x^2}\right), & \text{for } 0 \le x \le \dfrac{m}{3}, 0 \le y \le n \\ B.exp\left(\dfrac{-\left(x-\left(\dfrac{m}{3}+\dfrac{m}{6}\right)\right)^2}{2\sigma_x^2}\right), & \text{for } \dfrac{m}{3} \le x \le 2\dfrac{m}{6}, 0 \le y \le n \\ A.exp\left(\dfrac{-\left(x-\left(2\dfrac{m}{3}+\dfrac{m}{6}\right)\right)^2}{0.2\sigma_x^2}\right), & \text{for } 2\dfrac{m}{3} \le x \le m, 0 \le y \le n \end{cases} \tag{3.6}$$

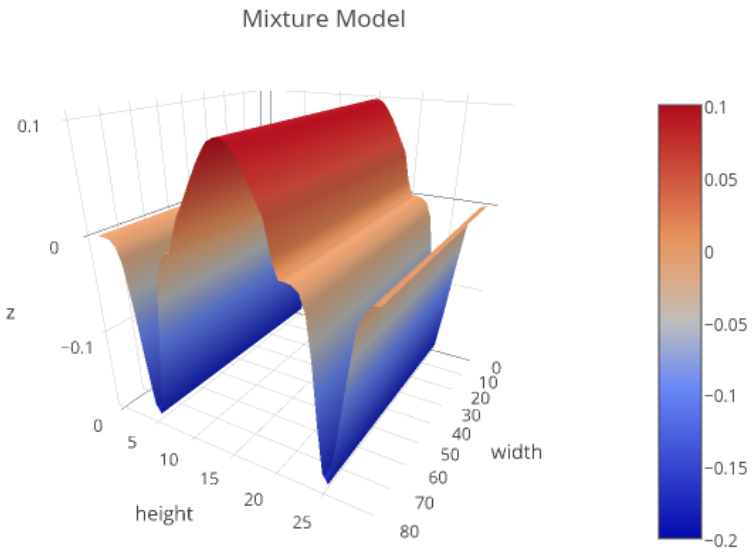


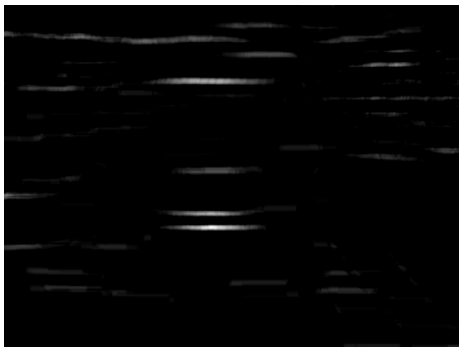

Figure 3.9: Gaussian Filter

In the Equation 3.6, $A$ and $B$ are coefficients of the mixture model. These parameters are set empirically following the condition: $A > 0$ and $B < 0$. The symbol $\sigma_x$ is the variance of the main lobe toward x direction. Figure 3.9 shows a plot depicting this equation.

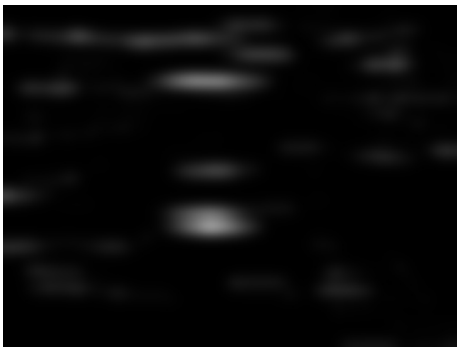
This filtering process provides a strong response at plate-like regions (Figure 3.10a). The result is compared against a predefined threshold value, which is around 80% of the maximum intensity. We call this process as smoothing, and the threshold value as *smoothing cutoff*.

### 3.3.7 Plate and regions extraction

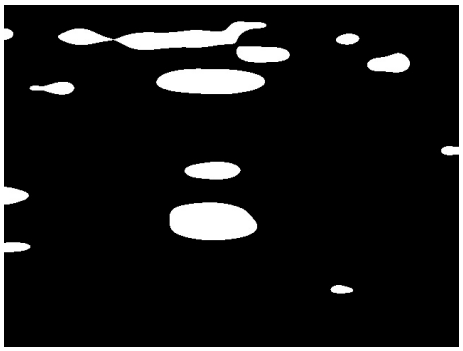
This step extracts the license plate and its bounding regions primarily. This does not detect actual license plate. But gives a close approximation to the location of the license plate. First we calculated all contours of the image calculated by mixture model. Next for each contours, we validated the bounding rectangle it is in. If the size of the bounding rectangle is valid, we extract the image of the area and region data. The extracted plates from the previous stage is shown in Figure 3.11.



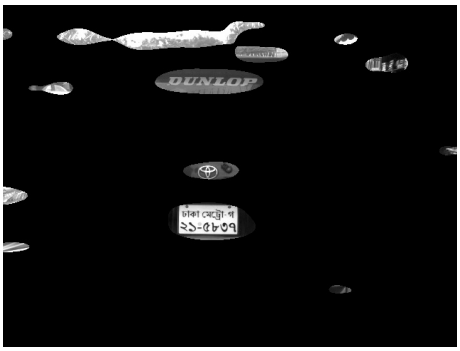
(a) After applying the mixture model



(b) Gaussian blur on 3.10a



(c) After applying adaptive threshold



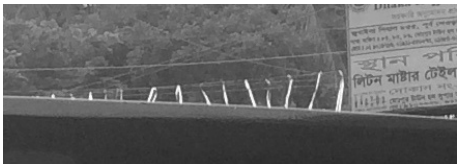

(d) A see-through view of 3.10c with main image

Figure 3.10: Images after applying mixture model.


(a) First estimation

(b) Second estimation


(c) Third estimation



(d) Fourth estimation

Figure 3.11: Estimated license plates.

*put an image here*

Figure 3.12: An overview of the plate detection procedure.


Figure 3.13: Applying threshold before Canny edge detection.

## 3.4 Plate Detection

This section describes the procedure to localize the license plate using the outputs from preprocessing stage. Figure 3.12 gives an overview on the entire method.

### 3.4.1 Edge Analysis

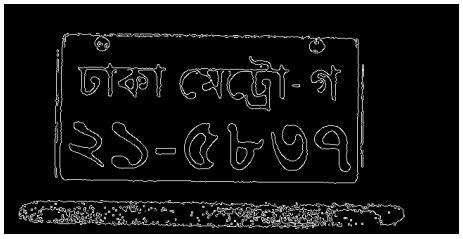
Before detectining edges we first applied a threshold value to the estimated license plates (Figure 3.13).

Canny Edge Detection is a popular edge detection algorithm. OpenCV has a direct function to it. We used the default function OpenCV provides to detect all edges of the localized estimated license plate image (Figure 3.14).

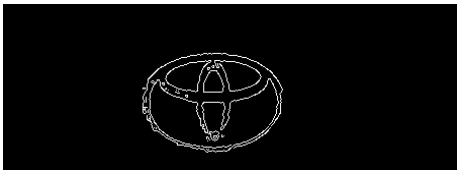
```
canny = cv2.Canny(img, 100, 200, L2gradient=True)
```

### 3.4.2 Contour Analysis

Canny edge detection works perfectly for most cases and provides an easy to detect contours around plate borders. After passing the canny image to the OpenCV's $findContours$ function, we get a set of contours. For each of these contours several parameters are checked before selecting a contours as a possible plate.




(a) Edges of 3.11a



(b) Edges of 3.11a


(c) Edges of 3.11a

(d) Edges of 3.11a

Figure 3.14: Result of Canny edge detecton algorithm.

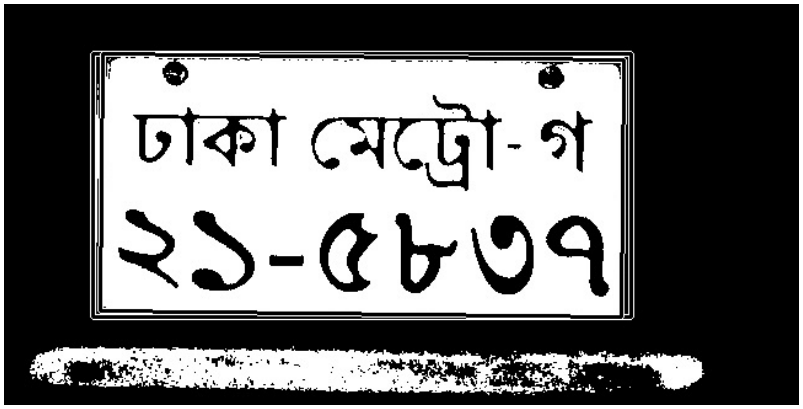



Figure 3.15: Only remaining candidate after contour analysis

In this steps, some of the estimated candidate from previous stages fall out. In Figure 3.15, all detected contours are highlighted.

#### 3.4.2.1 Width and Height

The width and height of the detected contour should be above a minimum margin to be recognizable by the OCR. We set the minimum width = 30 pixels and minimum height = 75 pixels for estimated plate.

#### 3.4.2.2 Area Ratio

Next we check if the area of the contour is greater than the 10% of the entire image. Remember, we got this image by localizing the plate like regions in pre-processing stage. A possible plate should not have the area below 10% of the entire image.

$$area\_ratio = \frac{contour\_area}{image\_area} \tag{3.7}$$

#### 3.4.2.3 Aspect Ratio

Equation 3.8 shows the calculation of aspect ratio of the contour image. From review and observations, we set the minimum value of aspect ratio to 0.3 and maximum to 0.6.

$$aspect\_ratio = \frac{contour\_height}{contour\_width} \tag{3.8}$$

#### 3.4.2.4 Rotation

The rotation can be found by the *minAreaRect* function from OpenCV.

```
angle = cv2.minAreaRect(cnt)[0]
```

If $10 \leq |angle| \leq 80$ we skipped the contour. Because otherwise the contour will be too much skewed to be recognized by any OCR.

### 3.4.3 Extraction

The region information we get from the previous step is used here to extract the plate image from the original image. We also rotate the image to match with the rotation angle found in previous step. This rotation is done by calculating a rotation matrix of scale 1 using a OpenCV function:

```
# Calculating rotation matrix M
M = cv2.getRotationMatrix2D((center_y, center_x), rotation, 1.0)
```

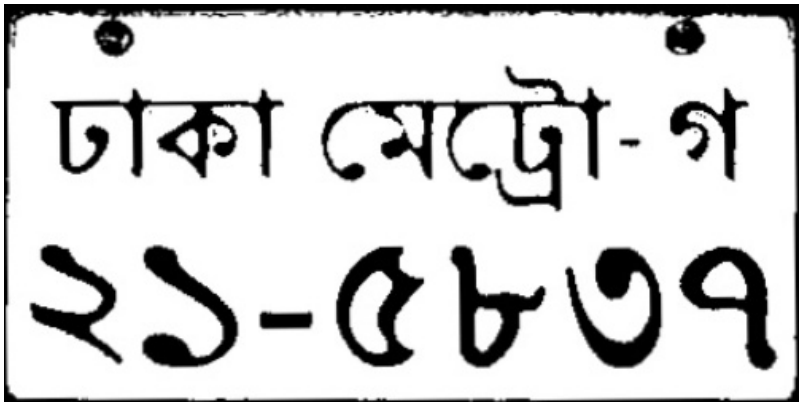



Figure 3.16: The extracted plate after rotation and scaling

The variable $rotation$ here is calculated from the $minAreaRect$'s $angle$ using Equation 3.9.

$$rotation = \begin{cases} -angle, & \text{if } angle \leq 45 \\ 90 - angle, & \text{if } angle > 45 \end{cases} \tag{3.9}$$

For convenience, the extracted plate image is rescaled into a dimension of $500 \times 250$. Figure 3.16 shows a sample.

### 3.4.4 Converting to Binary Image

The image from previous stage is need to be converted in binary for convenience. Bangladeshi license plates have two types: white text on black plate for private cars and black text on white plate for public ones. To be accurate, we first converted the image into black and white using two filters: $THRESH_BINARY$ one time and $THRESH_BINARY_INV$ the other. Then compared the ratio of non-zero pixels of each image. The image which has lesser ratio, most probably is selected as the expected binary image.

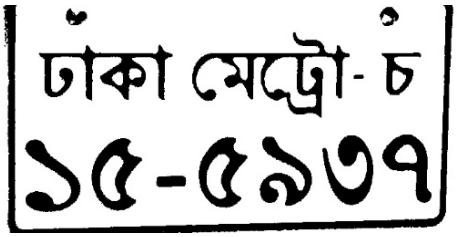
Figure 3.17 shows the conversion to binary images using the above algorithm for different types of license plates.

### 3.4.5 Denoising

De-noising is done in two parts: Removing the border lines, and removing the small contours. Figure 3.18 shows the effects of this algorithm.


(a) A private license plate


(b) Black and white image of 3.17a




(c) A public license plate


(d) Black and white image of 3.17c

Figure 3.17: Black and white conversion of two types of image

#### 3.4.5.1 Border Removal

Removing the borders of an image is tough job due to the diversity of plates and their origins. In our methods, we first removed the top, bottom, left and right border using OpenCV's $floodFill$ function. We checked all the pixels of first 20 rows and applied flood-fill on any non-black pixels. Similarly we check bottom 5 rows, from left first 12 columns and from right the first 12 columns.

#### 3.4.5.2 Contour cleaning

Only removing border is not enough to completely clear the plate image. We used contour analysis to blackout the remaining dots in the image that could not possibly be a letter. To find contour, we used $cv2.RETR_EXTERNAL$ mode which only wraps the external area (Otherwise the holes inside numbers and letters would be detected, and it would pose a problem).

After some observation, we set the minimum character height to be between 35 to $height - 30$ pixels and width between 35 to $width - 30$. We checked for all contours if it a possible character. If not, we used OpenCV's $fillConvexPoly$ method on $minAreaRect$ of the contour. It nicely fill up the contour with black pixels and hide it without affecting rest of the plate image.


(a) Noisy image (b) Removed borders

(c) Clean image

Figure 3.18: After applying Border removal and contour cleaning

*include some graphics*

Figure 3.19: Horizontal projection of a plate.

## 3.5 Segmentation

The goal of this stage is to separate all characters into different image to send them to OCR for recognition. We used horizontal and vertical projections in this step.

### 3.5.1 Horizontal Segmentation

Horizontal projection is done my taking mean of all columns across the rows of the image. Figure 3.19 shows a graph of mean values across the rows. To calculate this mean we followed the formula from Equation 3.10.

$$H = \sum_{i=1}^{height} \frac{\sum_{j=1}^{width} I_{ij}}{width} \tag{3.10}$$

We set the cutoff line to be $\geq 1.$ and separated the entire plate into two segments (Showing in

(a) Upper Segment

(b) Lower Segment

*include*

*two horizontal segments*

Figure 3.20: Horizontal segments of a plate.

Figure 3.20.

### 3.5.2 Vertical Segmentation

## 3.6 Character Recognition

# References

# Appendices