Vehicle Number Plate Detection

Project Report Submitted
for the Degree of
Bachelor In Technology
in
Computer Science

by

Shikhar Gupta

Shivam Darmora

Rishabh Gupta

CSE & IT DEPARTMENT
MS. RISHIKA YADAV
GRAPHIC ERA HILL UNIVERSITY, DEHRADUN,
UTTARAKHAND, INDIA

January,2020

The purpose of this report is to explain the implementation of our project, —Vehicle Number Plate Detection“. This report will begin with sections on motivation, past projects, and constraints. It will then proceed to describe our system in broad terms to provide a general overview of our project to the reader. It will then describe each subsystem in detail. For each subsystem, we have included explanations for why we chose our methods, performance of our methods, under what conditions would our methods fail, and how can we improve our methods.

The report ends with a discussion on possible future work using more evolved image processing techniques and acknowledgements.

Introduction

Vehicle Number Plate Detection (VNPD) is a part of digital image processing which is generally used in vehicle transportation system to categorize the vehicle. Number plate recognition systems are having varieties of application such as traffic maintenance, tracing stolen cars, automatic electronic Toll collection system etc. But the main aim is to control the traffic management system. In India the traffic management system is developing day by day. Monitoring vehicles for law enforcement and security purposes has become a difficult problem because of the number of automobiles on the road today.

In India, the number plate containing white background with black foreground color is used for private cars and for the commercial vehicles yellow is used as background and black as foreground color. The number plate starts with two digit letter “state code” followed by two digit numeral, followed by single letter after those four following digits .

There must exist a way for detecting and identifying license plates without constant human intervention. As a solution, we have implemented a system that can extract the license plate number of a vehicle from an image œ given a set of constraints.

Vehicle number plate detection aims at detection of vehicle license plate and then extracting the contents of the license plate i.e.(vehicle license number).

This data is collected and stored in a database, excel sheet , csv file etc.

Further the data (vehicle license numbers) that has been collected can be used for analysis based on various parameters which will provide best possible predictions.

These predictions will be helpful for an organization or a team to improve their work efficiency.

Motive

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection . Object detection has applications in many areas of computer vision, including image retreival and video surveillance.

Vehicle number plate recognition is an exciting and challenging research topic of object detection from past few years. Number plates are of different shape, size and also have different color in different countries. In India the most common vehicle number plate used have yellow or while as background and black as foreground color. In this report we have made a system for localization of number plate for vehicles in India and segmented the numbers as to identify each number separately. We generally focus on two steps; one is to locate the number plate and second is to segment all the number and letters to identify each number separately and therefore retrieve the license number.

Past works

Many plate detection, segmentation algorithm have been proposed to implement VNPI system. Number plate detection algorithm is mainly categorized into three classes: edge-based, color based and texture based. License plate location algorithm based on edge Detection and morphology are describe to locate the number plate, first identify whether any noise is present in the plate. Several segmentation and recognition methods are used for number plate segmentation. More correct and effective segmentation of number plate will produce virtuous and more efficient recognition. Based on the above mentioned technique, many number plate localization algorithms have been established.an upgraded and efficient approach is recognized with high detection rate based on sobel edge detection and morphological operation.

The Constraints

Due to the limited amount of time we have, a set of constraints have been placed on our system to make the project more manageable, they are as follows:

- Image of the vehicle taken from fixed distance.
- Image of the vehicle taken from fixed angle.
- Vehicle is stationary when the image was taken.

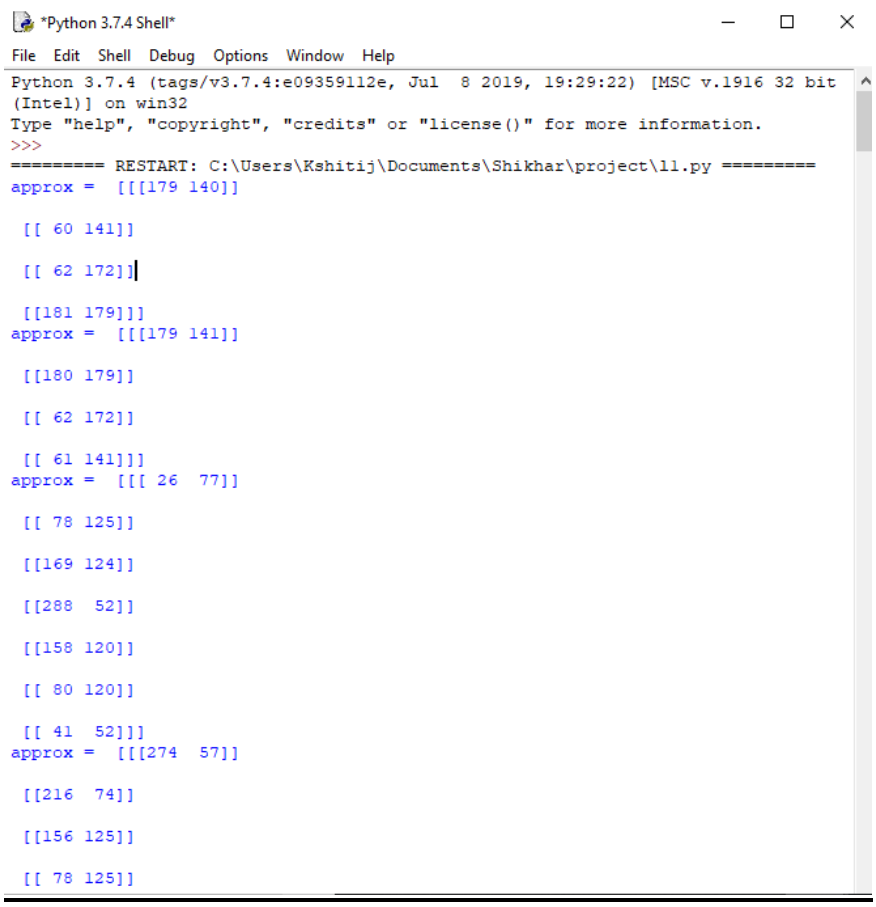
Source code

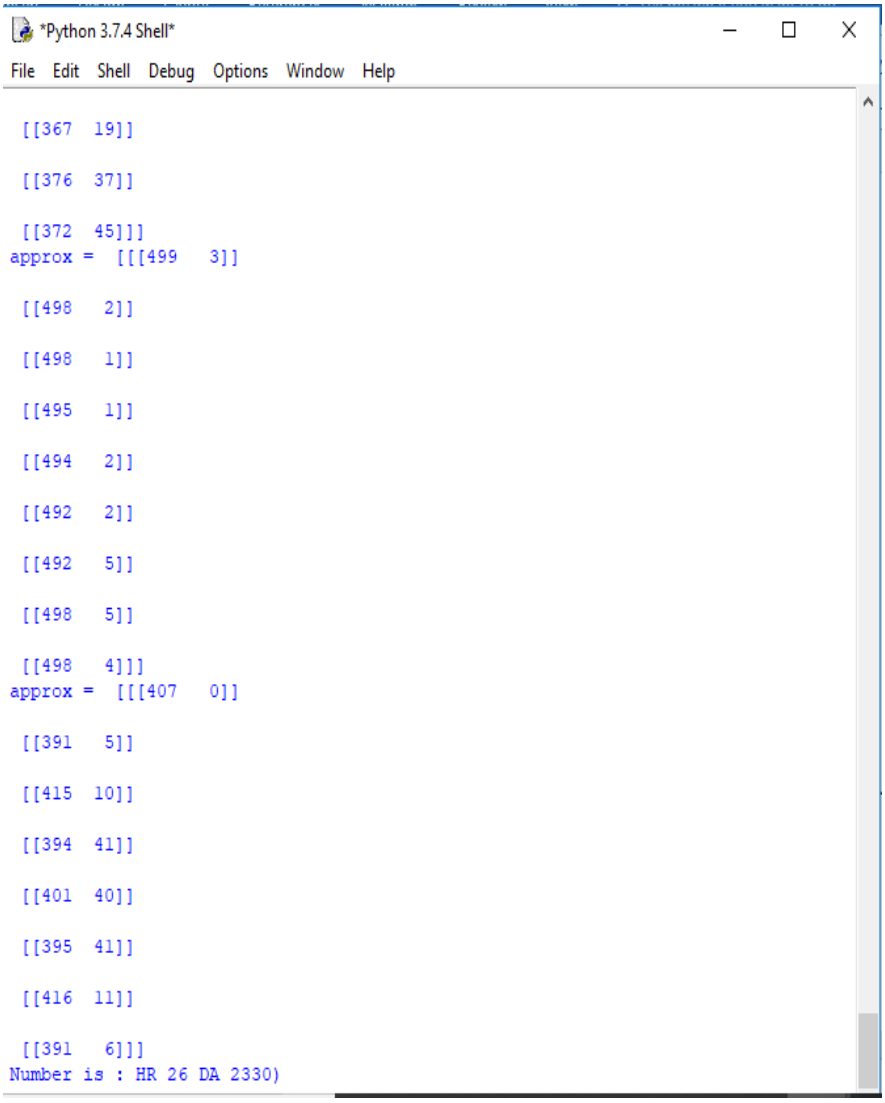
```
import imutils
import cv2
import numpy as np
import pytesseract as tess
tess.pytesseract.tesseract_cmd=r'C:\Program Files\Tesseract-OCR\tesseract.exe'

image=cv2.imread('4.jpeg')
image=imutils.resize(image, width=500)

cv2.imshow("Original Image",image)
cv2.waitKey(0)

gray=cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("1 - Grayscale Conversion", gray)
cv2.waitKey(0)

gray= cv2.bilateralFilter (gray, 11, 17, 17)
cv2.imshow("2 - Bilateral Filter", gray)
cv2.waitKey(0)

edged = cv2.Canny(gray, 170, 200)
cv2.imshow("3 - Canny Edged", edged)
cv2.waitKey(0)

cnts, new = cv2.findContours(edged.copy() , cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
```

```
mg1 = image.copy()
cv2.drawContours(img1, cnts, -1, (0,255,0), 3)
cv2.imshow("4- All Contours", img1)
cv2.waitKey(0)

cnts=sorted(cnts, key = cv2.contourArea, reverse = True)[:50]
NumberPlateCnt = None

img2 = image.copy()
cv2.drawContours(img2, cnts, -1, (0,255,0), 3)
cv2.imshow("5- Top 50 Contours", img2)
cv2.waitKey(0)

count = 0
idx = 7
for c in cnts:
      peri = cv2.arcLength(c, True)
      approx = cv2.approxPolyDP(c, 0.02 * peri, True)
      print("approx = ", approx)
     # if len(approx) == 4:
      NumberPlateCnt = approx #this is approx Number plate Countour
      x, y, w, h =cv2.boundingRect(c)
      new_img = image[y:y +h, x:x + w]
      cv2.imwrite('lp3/' + str(idx) + '.jpeg', new_img)
      idx+=1

cv2.drawContours(image, [NumberPlateCnt], -1, (0,255,0), 3)
```

```
cv2.imshow("Final Image With Number Plate Detected", image)
cv2.waitKey(0)



Cropped_img_loc = 'lp3/7.jpeg'
cv2.imshow("Cropped Image ", cv2.imread(Cropped_img_loc))

text = tess.image_to_string(Cropped_img_loc, lang='eng')
print("Number is :",text)

cv2.waitKey(0)
```

OUTPUT


Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:\Users\Kshitij\Documents\Shikhar\project\11.py =========
approx = [[[179 140]]
[[60 141]]
[[62 172]]
[[181 179]]]
approx = [[[179 141]]
[[180 179]]
[[62 172]]
[[61 141]]]
approx = [[[26 77]]
[[78 125]]
[[169 124]]
[[288 52]]
[[158 120]]
[[80 120]]
[[41 52]]]
approx = [[[274 57]]
[[216 74]]
[[156 125]]
[[78 125]]


Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
[[367 19]]
[[376 37]]
[[372 45]]]
approx = [[[499 3]]
[[498 2]]
[[498 1]]
[[495 1]]
[[494 2]]
[[492 2]]
[[492 5]]
[[498 5]]
[[498 4]]]
approx = [[[407 0]]
[[391 5]]
[[415 10]]
[[394 41]]
[[401 40]]
[[395 41]]
[[416 11]]
[[391 6]]]
Number is : HR 26 DA 2330)

Main Softwares

Open-cv :-

Open-cv is a type of computer software in which source code is released under a license in which the copyright holder grants users the rights to study, change, and distribute the software to anyone and for any purpose. Open-source software may be developed in a collaborative public manner. Open-source software is a prominent example of open collaboration.

Functions

- **cv2.imread():-**

  Method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.
  Syntax: cv2.imread(path, flag)
  Parameters:
  path: A string representing the path of the image to be read.
  flag: It specifies the way in which image should be read. It's default value is cv2.IMREAD_COLOR.

- **cv2.imshow():-**

  Method is used to display an image in a window. The window automatically fits to the image size.
  Syntax: cv2.imshow(window_name, image)
  Parameters:
  window_name: A string representing the name of the window in which image to be displayed.
  image: It is the image that is to be displayed.

- **cv2.findContour()**:-

  Function that helps in extracting the contours from the image. It works best on binary images, so we should first apply thresholding techniques, Sobel edges, etc.

- **cv2.drawContours()**:-

  It can also be used to draw any shape provided you have its boundary points. Its first argument is source image, second argument is the contours which should be passed as a Python list, third argument is index of contours (useful when drawing individual contour. To draw all contours, pass -1) and remaining arguments are color, thickness etc.

- **cv2.cvtColor():-**

  Method is used to convert an image from one color space to another. There are more than 150 color-space conversion methods available in OpenCV. We will use some of color space conversion codes below.

- **cv.bilateralFilter():-**

  It is highly effective in noise removal while keeping edges sharp. But the operation is slower compared to other filters. We already saw that a Gaussian filter takes the neighbourhood around the pixel and finds its Gaussian weighted average. This Gaussian filter is a function of space alone, that is, nearby pixels are considered while filtering. It doesn't consider whether pixels have almost the same intensity. It doesn't consider whether a pixel is an edge pixel or not. So it blurs the edges also, which we don't want to do.

- **cv2.Canny():-**

  Which takes our input image as first argument and its aperture size(min value and max value) as last two arguments. This is a simple example of how to detect edges in Python.

- **cv2.arcLength():-**

  Calculates a contour perimeter or a curve length.

  Python**:** cv2.arcLength(curve, closed)

Pytesseract

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images.

Python-tesseract is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

Functions

- **tess.image_to_string():-**
  Our goal is to convert a given text image into a string of text, saving it to a file and to hear what is written in the image

Imutils

Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python

Functions

- **imutils.resize():-**

  It calculates the width and height of template in w and r, anf initialize a variable found to keep track of the region and scale of the image with the best match.From the location where is stored in the computer.

Steps of working

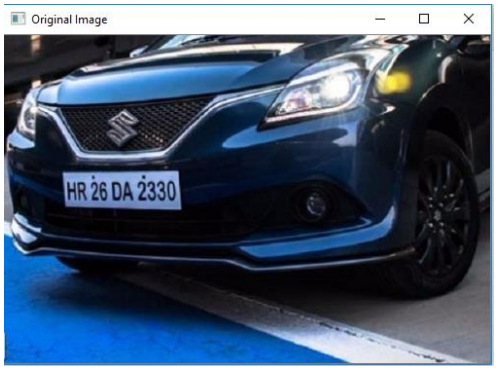
STEP1: Grayscale Conversion

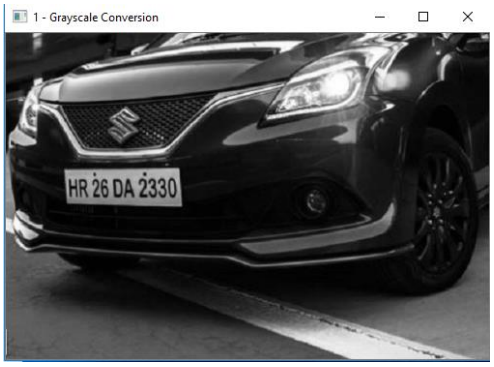
A grayscale or greyscale image is one in which the value of each pixel is a single sample representing only an amount of light, that is, it carries only intensity information. Grayscale images, a kind of black-and-white or gray monochrome, are composed exclusively of shades of gray. The contrast ranges from black at the weakest intensity to white at the strongest.

In and color image by emitting a controlled combination of these three basic colors (red,green and blue) we are able to generate almost any perceivable color. This is the reasoning behind why color images are often stored as three separate image matrices; one storing the amount of red (R) in each pixel, one the amount of green (G) and one the amount of blue (B). We call such color images as stored in an RGB format.

In grayscale images, however, we do not differentiate how much we emit of the different colors, we emit the same amount in each channel. What we can differentiate is the total amount of emitted light for each pixel; little light gives dark pixels and much light is perceived as bright pixels.

In this step the original image is converted into a grayscale image.

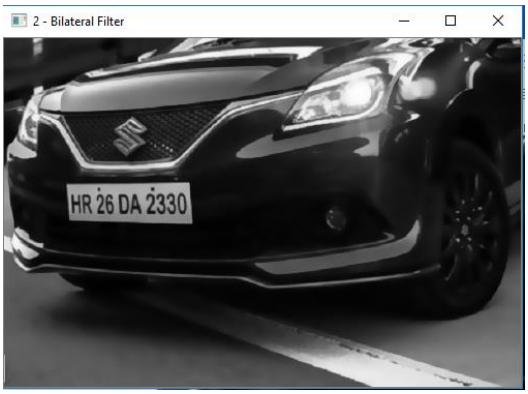




STEP2: Bilateral Filter

A bilateral filter is a non-linear, edge-preserving, and noise-reducing smoothing filter for images. It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels. This weight can be based on a Gaussian distribution. Crucially, the weights depend not only on Euclidean distance of pixels, but also on the radiometric differences (e.g., range differences, such as color intensity, depth distance, etc.). This preserves sharp edges.

The basic idea underlying bilateral filtering is to do in the range of an image what traditional filters do in its domain. Two pixels can be close to one another, that is, occupy nearby spatial location, or they can be similar to one another, that is, have nearby values, possibly in a perceptually meaningful fashion.

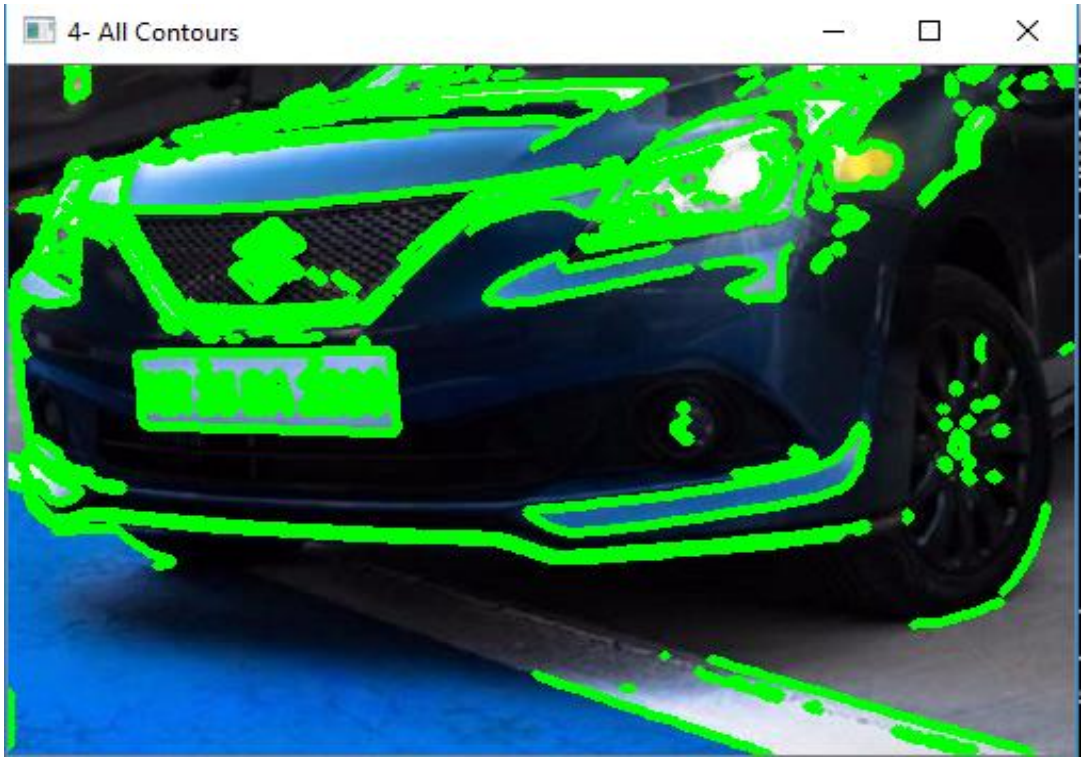


STEP3: Canny Edged

Canny edge detection is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed. It has been widely applied in various computer vision systems. Canny has found that the requirements for the application of edge detection on diverse vision systems are relatively similar. Thus, an edge detection solution to address these requirements can be implemented in a wide range of situations. The general criteria for edge detection include:

1. Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible
2. The edge point detected from the operator should accurately localize on the center of the edge.
3. A given edge in the image should only be marked once, and where possible, image noise should not create false edges

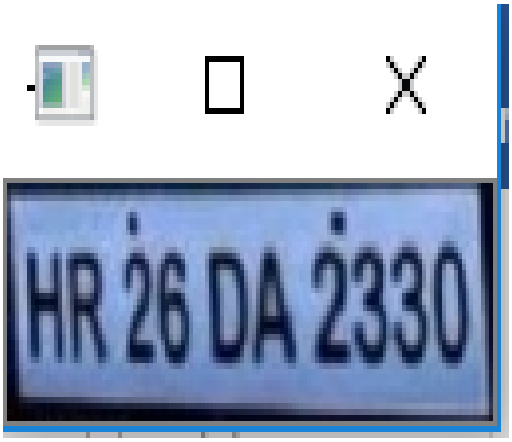
STEP4: All Contours



STEP5: Top 50 Contours

STEP6: Detection of number plate

Step 7: Image after detection.



Step 8: Extraction of number.

```
 [[391   6]]]
Number is : HR 26 DA 2330)
```

Future Work

Due to time constraints and the lack of experience in image processing in our group, we are unable to make this license plate recognition system as functional as it could be. There are numerous improvements that could be made, such as:

• Instead of breaking out of the program after character segmentation fails (which usually indicate that we have chosen the wrong candidate), the program should have a system of selecting the next best candidate and continue to perform OCR on successive candidates.

• Use more evolved image processing techniques to improve the accuracy of the system. For example, we could use binary morphology to eliminate edges that are thinner than the characters of the license plate.

• Increase the resolution of the images.

• Expand the system to work with variable angle and distance.

• Expand the system to work with various License Plates.