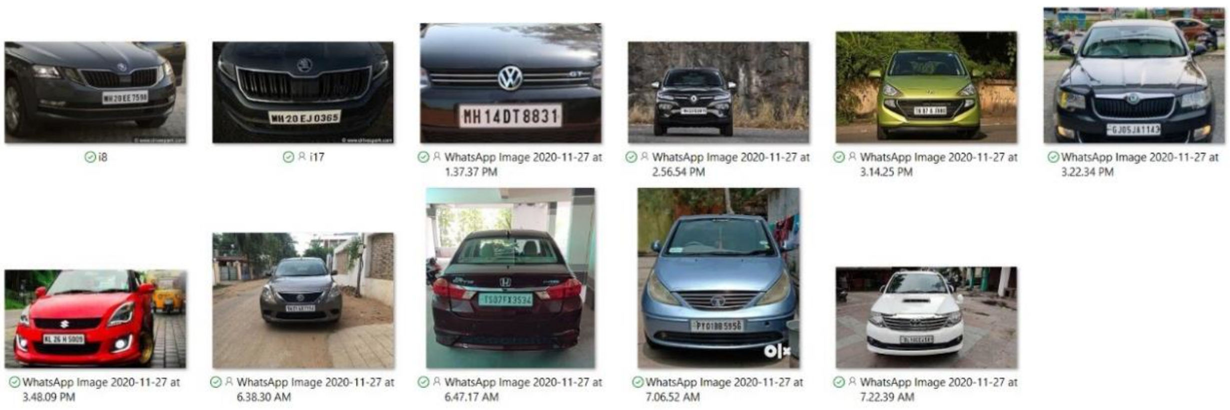
Table of contents:

Abstract:

Number Plate recognition, also called License Plate realization or recognition using image processing methods is a potential research area in smart cities and the Internet of Things. An exponential increase in the number of vehicles necessitates the use of automated systems to maintain vehicle information for various purposes. In the proposed algorithm an efficient method for recognition of Indian vehicle number plates has been devised. We are able to deal with noisy, low illuminated, cross angled, non-standard font number plates. This work employs several image processing techniques such as, morphological transformation, Gaussian smoothing, Gaussian thresholding and Sobel edge detection method in the pre-processing stage, afterwhich number plate segmentation, contours are applied by border following and contours are filtered based on character dimensions and spatial localization. Finally we apply Optical Character Recognition (OCR) to recognize the extracted characters. The detected texts are stored in the database, further which they are sorted and made available for searching. The project has its own drawbacks and limitations as we are not using higher machine learning or deep learning algorithms but it works efficiently for an average use case.

Requirements:

Dataset: Images of Indian vehicles where the number plate is clearly visible.


i8
i17
WhatsApp Image 2020-11-27 at 1.37.37 PM
WhatsApp Image 2020-11-27 at 2.56.54 PM
WhatsApp Image 2020-11-27 at 3.14.25 PM
WhatsApp Image 2020-11-27 at 3.22.34 PM
WhatsApp Image 2020-11-27 at 3.48.09 PM
WhatsApp Image 2020-11-27 at 6.38.30 AM
WhatsApp Image 2020-11-27 at 6.47.17 AM
WhatsApp Image 2020-11-27 at 7.06.52 AM
WhatsApp Image 2020-11-27 at 7.22.39 AM

Operating System: Windows 10

Programming language: Python3

Libraries used: sys, glob, os, glob, numpy, cv2, PIL, pytesseract, re

Applications:

Automatic Number Plate Recognition (ANPR) has a wide range of applications since the license number is the primary, most widely accepted, human readable, mandatory identifier of motor vehicles. Some of the applications are:

1. **Housing societies / Apartments:** ANPR can be used in housing societies to let in resident's vehicles inside by storing the number plate details in the database. This can hence reduce the man-force near the security gate. Our project works efficiently in this case provided the society or apartment is not very large.
2. **Parking:** Ticketless parking fee management, parking access automation, vehicle location guidance, parking fee charging, car theft prevention, "lost ticket" fraud, fraud by changing tickets are some areas where ANPR is helpful .
3. **Access Control:** License plate recognition brings automation of vehicle access control management, providing increased security, car pool management for logistics, security guide assistance, event logging, event management, keeping access diary, possibilities for analysis and data mining. It is very effective in Border and Law controls too.
4. **Motorway Road Tolling:** Tolls are a common way of funding the improvements of highways, motorways, roads and bridges. Efficient road tolling reduces fraud related to non-payment, makes charging effective, reduces required manpower to process events of exceptions and these can be implemented using ANPR.
5. **Journey Time Measurement:** Data collected by license plate recognition systems can be used in many ways after processing, feeding back information to road users to increase traffic security, helping efficient law enforcement, optimising traffic routes and reducing costs and time.

Methodology:

The proposed methodology consists of four major phases: pre-processing, detection, recognition and searching as shown in figure below.

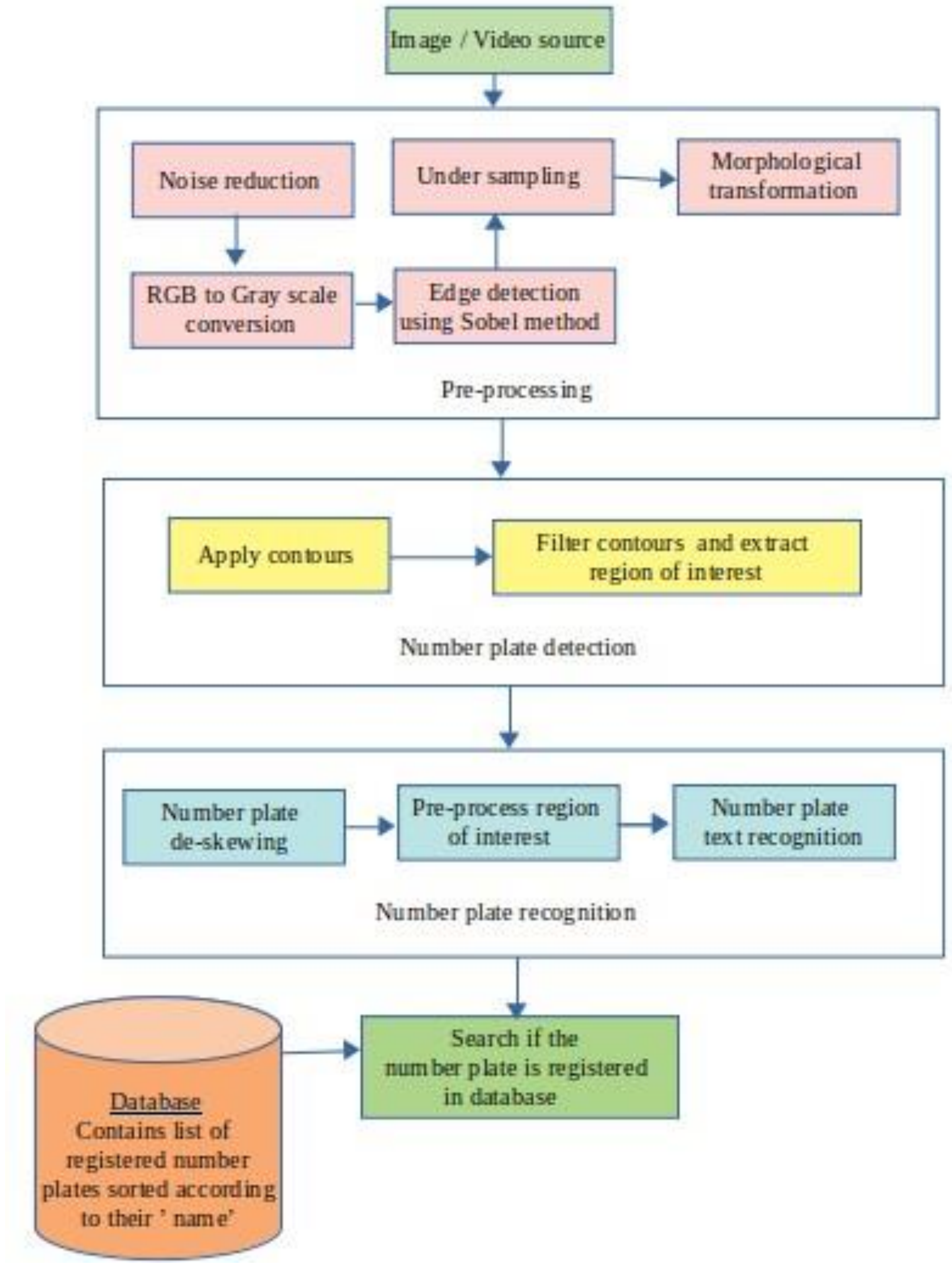


Image / Video source
Noise reduction
Under sampling
Morphological transformation
RGB to Gray scale conversion
Edge detection using Sobel method
Pre-processing
Apply contours
Filter contours and extract region of interest
Number plate detection
Number plate de-skewing
Pre-process region of interest
Number plate text recognition
Number plate recognition
Database
Contains list of registered number plates sorted according to their ' name'
Search if the number plate is registered in database

Brief description of steps:

Step 1: Image pre-processing

Step 1.1: Noise reduction: The objective of Gaussian filtering/ Gaussian smoothing is to reduce noise and detail. This will serve well for further image processing steps. For an Image, mathematically, gaussian filter can be expressed as:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

The input image is made to convolve with this 2-D 'G' matrix to obtain a smoothened image. In OpenCV, Gaussian smoothing can be applied using the following function: $cv2.GaussianBlur(image,(5,5),0)$, where (5,5) refers to the filter size and '0' indicates the model to find the value of standard deviation (sigma) itself.

Step 1.2: RGB to Grayscale conversion: Converting RGB image to grayscale saves a lot of time since we have to perform convolution of the image with sobel filter over only one 2D matrix rather than RGB image having 3 channels and making it complicated. Another reason is, in case of image edge detection we are focussed on observing the intensity change and it is easier to analyse it in a gray-scaled image.

Step 1.3: Edge detection using sobel method: Sobel edge detection works by calculating the gradient of image intensity at each pixel within the image. It finds the direction of the largest increase from light to dark and the rate of change in that direction. The following terms are used to perform edge detection using Sobel method:

Run filter over image

$$\frac{\partial f}{\partial x} = S_x \otimes f \qquad \frac{\partial f}{\partial y} = S_y \otimes f$$

Image gradient

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

In OpenCV, cv2.Sobel(img2,cv2.CV_8U,1,0,ksize=3) is used to perform edge detection using kernel size of 3.

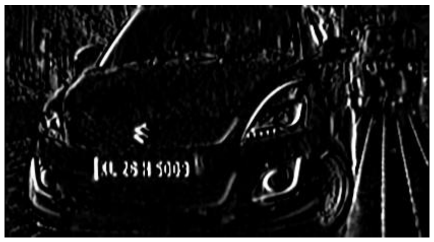
Step 1.4: Under-sampling: The Number plate detection and recognition algorithm are supposed to work at a steady and consistent frame rate. Unsurprisingly, for high-resolution images, image processing algorithms tend to work slow. It is in fact unnecessary to consider images with such a high resolution. This stage reduces the resolution if it crosses a predefined threshold.

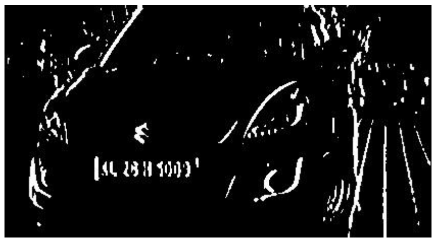
Step 1.5: Morphological transformation: Top-hat and Black-hat filters are part of Morphological transformations . The Top-hat operation is used to enhance bright objects of interest in a relatively dark background, while the black-hat operation (also known as bottom-hat) is used to enhance dark objects of interest in a relatively bright background. In this work, top-hat results are added to the original image and black-hat results are subtracted from it.

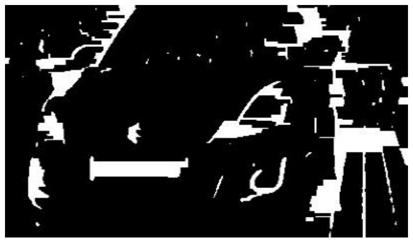
(New image)

(Step 1.1)

(Step 1.2)


(Step 1.3)

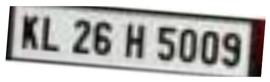

(Step 1.4)


(Step 1.5)

Step 2: Number plate detection

Step 2.1: Apply Counters: Contour Tracing, also called as Border following is the algorithm used for generating Contours. A contour is a link of equal intensity points along the boundary. In OpenCV, finding contours is like finding a white object from the black background, therefore during the Adaptive Gaussian Thresholding stage, Inversion operation has to be applied.

Step 2.2: Filter Contours and extract region of interest: For small regions, especially sharp edges and noise outliers, contours are applied. A human eye can easily figure out that such contours are unnecessary, but this must be incorporated into a program. Initially, Bounding boxes were applied to each contour. Then, for each contour, the following factors were considered such as minimum contour area, minimum contour width and height, minimum and maximum possible aspect ratios. This resulted in the filtering of most of the unnecessary contours, propelling us near to our objective, ie, Detect number plate.



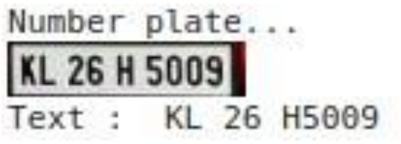
(Step 2)

Step 3: Number plate recognition

Step 3.1: Number plate de-skewing: Skew is the amount of rotation necessary to return an image to horizontal and vertical alignment. Skew is measured in degrees. Deskewing is a process whereby skew is removed by rotating an image by the same amount as its skew but in the opposite direction. This results in a horizontally and vertically aligned image where the text runs across the page rather than at an angle. In our project, this step is done using ratio_and_rotation()

Step 3.2: Pre-process region of interest: It is possible that two or more contours may completely overlap with each other, as in the case with the number 'zero'. The inner contour, if detected in the contour process, may lie completely inside its outer contour. Due to this phenomenon, both contours may get recognized as separate characters during the recognition process. If needed, we also resize the image before doing the recognition step.

Step 3.3:Number plate text recognition: Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images. We have used this tool finally to obtain the text present in the filtered, de-skewed contour.



```
Number plate...
```

```
Text :  KL 26 H5009
```

(Step 3)

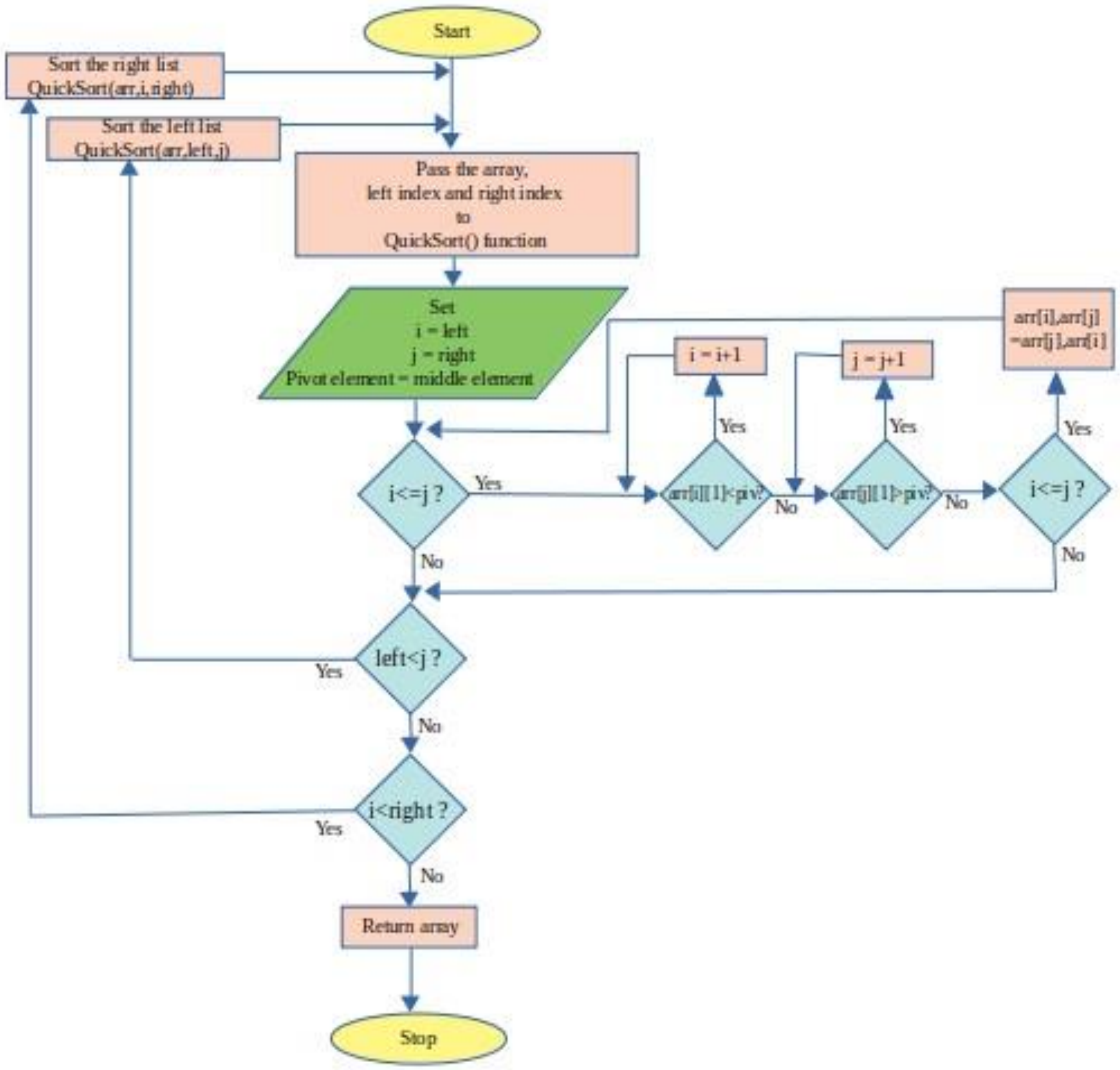
Step 4: Searching unknown image

Step 4.1: Create database: Using Step-1,2 and 3, register all the vehicles in the dataset and store them in a database after removing other special characters.

Step 4.2: Sorting: To make the final stage of searching more efficient, we are performing sorting operations on the detected texts. This is done using a quick sort algorithm. Quick sort is a divide and conquer algorithm. It is not stable and does in-place sorting.

Quick sort:

Flowchart:


Start
Sort the right list QuickSort(arr,i,right)
Sort the left list QuickSort(arr,left,j)
Pass the array, left index and right index to QuickSort() function
Set i = left j = right Pivot element = middle element
i = i+1
j = j+1
arr[i],arr[j] =arr[j],arr[i]
i<=j ?
arr[i][1]<piv?
arr[j][1]>piv?
i<=j ?
Yes
No
left<j ?
i<right ?
Return array
Stop

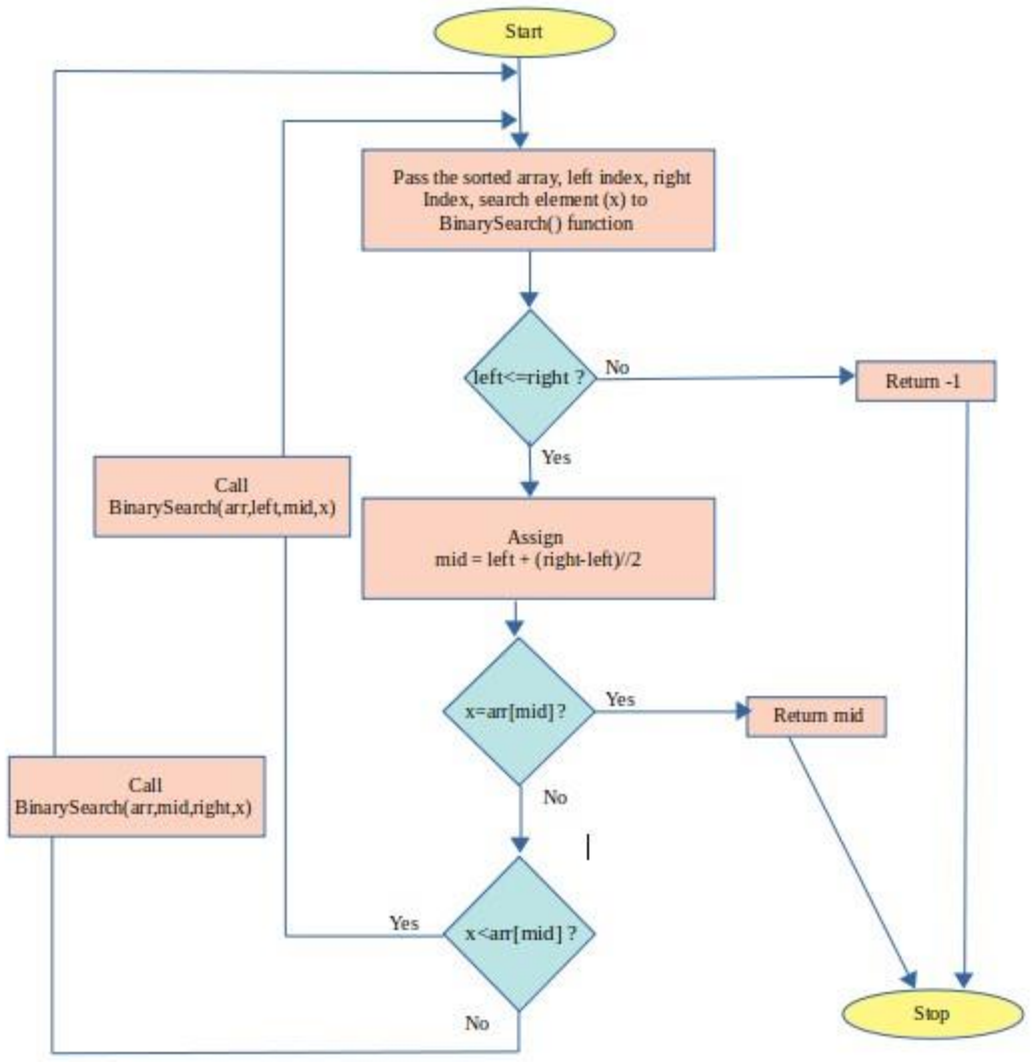
Time complexity:
Best case: O(n log(n))
Average case: O(n log(n))
Worst case: O(n^2)

Space complexity:
Best case: O(log(n))
Worst case: O(n)

Step 4.3: Searching : Pass a new image and follow steps 1,2,3. Obtain the new vehicle's registration number and check if it is present in the database using Binary search method. Binary search is another simple divide and conquer algorithm that is performed on a sorted array/list. It works better than linear search in case of more images in the dataset.

Binary search:

Flowchart:


Start
Pass the sorted array, left index, right Index, search element (x) to BinarySearch() function
left<=right ?
No
Return -1
Yes
Call BinarySearch(arr,left,mid,x)
Assign mid = left + (right-left)//2
x=arr[mid] ?
Yes
Return mid
No
Call BinarySearch(arr,mid,right,x)
Yes
x<arr[mid] ?
No
Stop

Time complexity:

It takes O(log(n)) time complexity, which when c ompared with linear search is much better, especially when the array size gets larger.

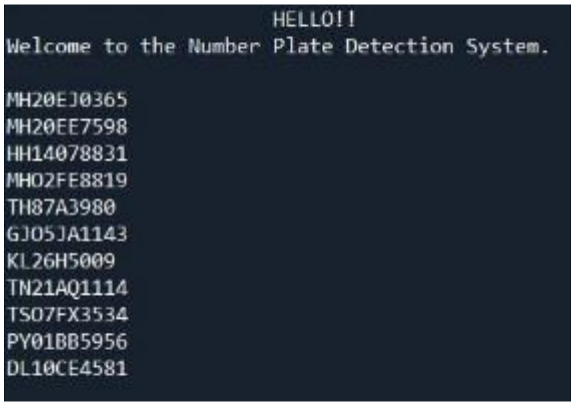
Space complexity:

O(1)

Program:-

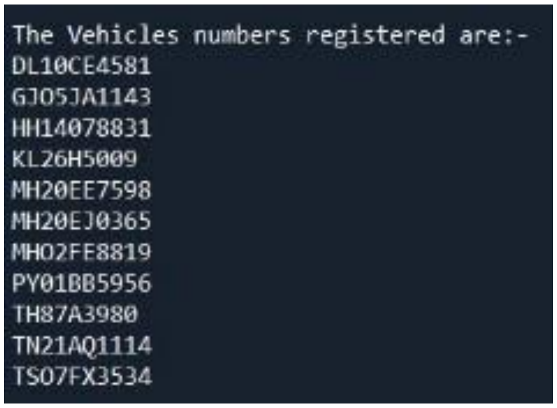
```
import sys
import glob
import os
import glob
import numpy as np
import cv2
from PIL import Image
import pytesseract
import re

#Detecting numberplate
def number_plate_detection(img):
    def clean2_plate(plate):
        gray_img = cv2.cvtColor(plate, cv2.COLOR_BGR2GRAY)

        _, thresh = cv2.threshold(gray_img, 110, 255, cv2.THRESH_BINARY)
        if cv2.waitKey(0) & 0xff == ord('q'):
            pass
        num_contours,hierarchy = cv2.findContours(thresh.copy(),cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)

        if num_contours:
            contour_area = [cv2.contourArea(c) for c in num_contours]
            max_cntr_index = np.argmax(contour_area)

            max_cnt = num_contours[max_cntr_index]
            max_cntArea = contour_area[max_cntr_index]
            x,y,w,h = cv2.boundingRect(max_cnt)

            if not ratioCheck(max_cntArea,w,h):
                return plate,None

            final_img = thresh[y:y+h, x:x+w]
            return final_img,[x,y,w,h]

        else:
            return plate,None

    def ratioCheck(area, width, height):
        ratio = float(width) / float(height)
        if ratio < 1:
            ratio = 1 / ratio
        if (area < 1063.62 or area > 73862.5) or (ratio < 3 or ratio > 6):
            return False
        return True

    def isMaxWhite(plate):
        avg = np.mean(plate)
        if(avg>=115):
            return True
        else:
            return False

    def ratio_and_rotation(rect):
        (x, y), (width, height), rect_angle = rect

        if(width>height):
            angle = -rect_angle
        else:
            angle = 90 + rect_angle

        if angle>15:
            return False

        if height == 0 or width == 0:
            return False

        area = height*width
        if not ratioCheck(area,width,height):
            return False
        else:
            return True


    img2 = cv2.GaussianBlur(img, (5,5), 0)
    img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

    img2 = cv2.Sobel(img2,cv2.CV_8U,1,0,ksize=3)
    _,img2 = cv2.threshold(img2,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

    element = cv2.getStructuringElement(shape=cv2.MORPH_RECT, ksize=(17, 3))
    morph_img_threshold = img2.copy()
    cv2.morphologyEx(src=img2, op=cv2.MORPH_CLOSE, kernel=element, dst=morph_img_threshold)
    num_contours, hierarchy=
  cv2.findContours(morph_img_threshold,mode=cv2.RETR_EXTERNAL,method=cv2.CHAIN_APPROX_NONE)
    cv2.drawContours(img2, num_contours, -1, (0,255,0), 1)


    for i,cnt in enumerate(num_contours):
        min_rect = cv2.minAreaRect(cnt)
        if ratio_and_rotation(min_rect):
            x,y,w,h = cv2.boundingRect(cnt)
            plate_img = img[y:y+h,x:x+w]
            if(isMaxWhite(plate_img)):
                clean_plate, rect = clean2_plate(plate_img)
                if rect:
                    fg=0
                    x1,y1,w1,h1 = rect
                    x,y,w,h = x+x1,y+y1,w1,h1
                    plate_im = Image.fromarray(clean_plate)
                    text = pytesseract.image_to_string(plate_im, lang='eng')
                    return text

#Quick sort
def partition(arr,low,high):
    i = ( low-1 )
    pivot = arr[high]

    for j in range(low , high):
        if   arr[j] < pivot:
            i = i+1
            arr[i],arr[j] = arr[j],arr[i]

    arr[i+1],arr[high] = arr[high],arr[i+1]
    return ( i+1 )

def quickSort(arr,low,high):
    if low < high:
        pi = partition(arr,low,high)

        quickSort(arr, low, pi-1)
        quickSort(arr, pi+1, high)

    return arr

#Binary search
def binarySearch (arr, l, r, x):

    if r >= l:
        mid = l + (r - l) // 2
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            return binarySearch(arr, l, mid-1, x)
        else:
            return binarySearch(arr, mid + 1, r, x)
    else:
        return -1


print("HELLO!!")
print("Welcome to the Number Plate Detection System.\n")

array=[]

dir = os.path.dirname(__file__)

for img in glob.glob(dir+"/Images/*.jpeg") :
    img=cv2.imread(img)

    img2 = cv2.resize(img, (600, 600))
    cv2.imshow("Image of car ",img2)
    cv2.waitKey(1000)
    cv2.destroyAllWindows()


    number_plate=number_plate_detection(img)
    res2 = str("".join(re.split("[^a-zA-Z0-9]*", number_plate)))
    res2=res2.upper()
    print(res2)

    array.append(res2)

#Sorting
array=quickSort(array,0,len(array)-1)
print ("\n\n")
print("The Vehicle numbers registered are:-")
for i in array:
    print(i)
print ("\n\n")

#Searching
for img in glob.glob(dir+"/search/*.jpeg") :
    img=cv2.imread(img)

    number_plate=number_plate_detection(img)
    res2 = str("".join(re.split("[^a-zA-Z0-9]*", number_plate)))

print("The car number to search is:- ",res2)
```

```
result = binarySearch(array,0,len(array)-1,res2)
if result != -1:
            print ("\n\nThe Vehicle is allowed to visit." )
else:
    print ("\n\nThe Vehicle is  not allowed to visit.")
```



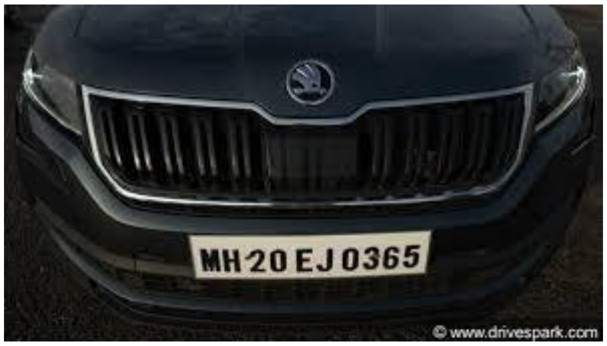
```
                          HELLO!!
Welcome to the Number Plate Detection System.

MH20EJ0365
MH20EE7598
HH14078831
MHO2FE8819
TH87A3980
GJ05JA1143
KL26H5009
TN21AQ1114
TSO7FX3534
PY01BB5956
DL10CE4581
```

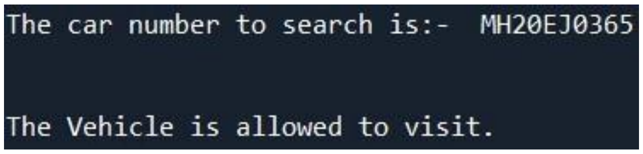
(Detected number plate)



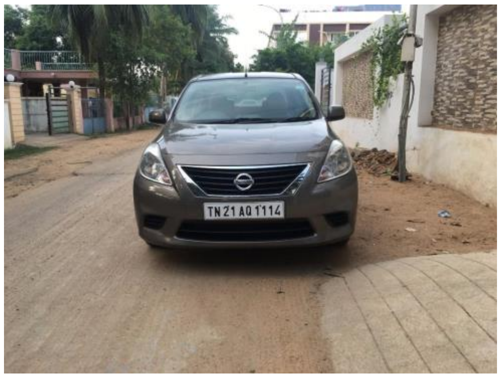
```
The Vehicles numbers registered are:-
DL10CE4581
GJ05JA1143
HH14078831
KL26H5009
MH20EE7598
MH20EJ0365
MHO2FE8819
PY01BB5956
TH87A3980
TN21AQ1114
TSO7FX3534
```

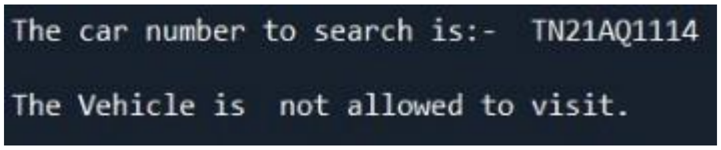
(Sorted number plates)





```
The car number to search is:-  MH20EJ0365

The Vehicle is allowed to visit.
```





```
The car number to search is:-  TN21AQ1114

The Vehicle is  not allowed to visit.
```

(New image)

(Searched - output)

Challenges and solution:

1. **Dealing with bright and dark objects.**
   Solution: Many details present in an RGB image b ecome less salient when we convert it into grayscale. To visualize the change in i ntensity and thus to deal with bright and dark objects in the image, converting it to grayscale works best.

2. **Dealing with noisy images.**
   Solution: We have convolved the image with a Gaussian filter in the pre-processing stage to reduce the noise present in the image.

3. **Dealing with cross-angled or skewed number plates.**
   Solution: De-skewing, i.e. rotating the i mage to the required position can be done t o detect the text present on the number plate.

4. **Dealing with non-standard number plates.**
   Solution:The code we implemented won't g ive any results in case of non-standard/ partially torn number plates. Hence, they won't be stored in the database and searching is not possible on it.

Conclusion:

Since we didn't use complex machine learning and deep learning algorithms there are some drawbacks of this project but it will work efficiently if implemented in housing society/apartment/institution to allow resident's vehicles inside and almost all the challenges we faced while solving the problem are resolved to a good extent.

References:

- M M Shidore, and S P Narote. (2011) "Number Plate Recognition for Indian Vehicles" International Journal of Computer Science and Network Security 1 1 (2): 143-146
- Sang Kyoon Kim, D. W. Kim and Hang Joon Kim. (1996) "A recognition of vehicle license plate using a genetic algorithm based segmentation," Proceedings of 3 rd IEEE International Conference on Image Processing, Lausanne.
- https://docs.opencv.org/master/