APARTMENT CAR ENTRY SYSTEM

PROJECT REPORT

Submitted for CAL in B.Tech Artificial Intelligence (CSE3013)

By

MALLADI TEJASVI 15BCE1208
P.S.N SURYAVAMSI 15BEC1127

Slot: D1+TD1

Name of Faculty: Dr.Bhargavi R(SCSE)

(SCHOOL OF COMPUTING SCIENCE AND ENGINEERING)

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

MARCH, 2018

CERTIFICATE

This is to certify that the Project work entitled " **Apartment Car Entry System"** that is being submitted by " ***Malladi Tejasvi, P.S.N Suryavamsi*** for CAL in B.Tech Artificial Intelligence(CSE3013) is a record of bonafide work done under my supervision. The contents of this Project work, in full or in parts, have neither been taken from any other source.

Place :Chennai

Date: 30/03/2018

Signature of Students: Malladi Tejasvi
P.S.N Suryavamsi

Signature Of Faculty: Dr. Bhargavi R

ACKNOWLEDGEMENTS

We acknowledge and immensely thank VIT University chennai and its management for providing us an opportunity to carry out this project.

We also would like to convey our sincere thanks to our Professor Dr. Bhargavi Rmadam, for her constant support and guidance.

Malladi Tejasvi
15BCE1208

P.S.N Suryavamsi
15BEC1127

ABSTRACT

Having a toll like car entry system, where all the cars that belong to the community are given a card to be swiped in front of a reader for the gate to open. Having such a system is desirable and beneficial interms of the increase in security achieved through allowing only authorized cars. The process of swiping a card is difficult for a driver because (s)he must be controlling the car so that it is ready to move as soon as the gate is open and in this process the driver needs to be careful to ensure that the car crosses before the gate closes and doesn't stop just under the closing gate and get hit by the gate. All these problems are solved if the gate is smart enough to find if a car belongs to the apartments, by itself and open the gate for the cars which belong to the community to have a hassle free and secure system.

1.0 INTRODUCTION

1.1 Objective and Goal of the project

To design and build a prototype of automatic apartment car entry system, that opens the car entrance gate, when a car of the aparmetnets enters the community.

1.2 System Overview

The system, when commercialized, is to be installed at an electronic gate at apartment car entry point. When a car is four meters away from the gate, the properly positioned and installed camera captures an image which is processed and the car number is extracted, then the database is searched if the car belongs to the community. If yes, the gate opens else an alert is generated. The existing systems have a electronic gate at the entry point which opens when the car driver, who would have a RFID based membership card and stops at the gate in a position that is close to the card reader and swipes the card and waits for the gate to open. In contrast to this system the proposed system automatically recognizes the presence of a car and finds the car number and whitout any hassel it lets the car in. The proposed system uses a novel way of the car detection-using ultrasonic sensor, instead of object detection using image processing which is computationally expensive. The algorithm used for segmenting number plate is k-means clustering followed by grab and cut algorithm , this is a novel approach for number plate identification and such a method was not followed in any relavents research paper. This approach is feasible, keeping in mind the computational limitation of an embedded system.

2.0 METHODOLOGY

2.1 Hardware Components Used

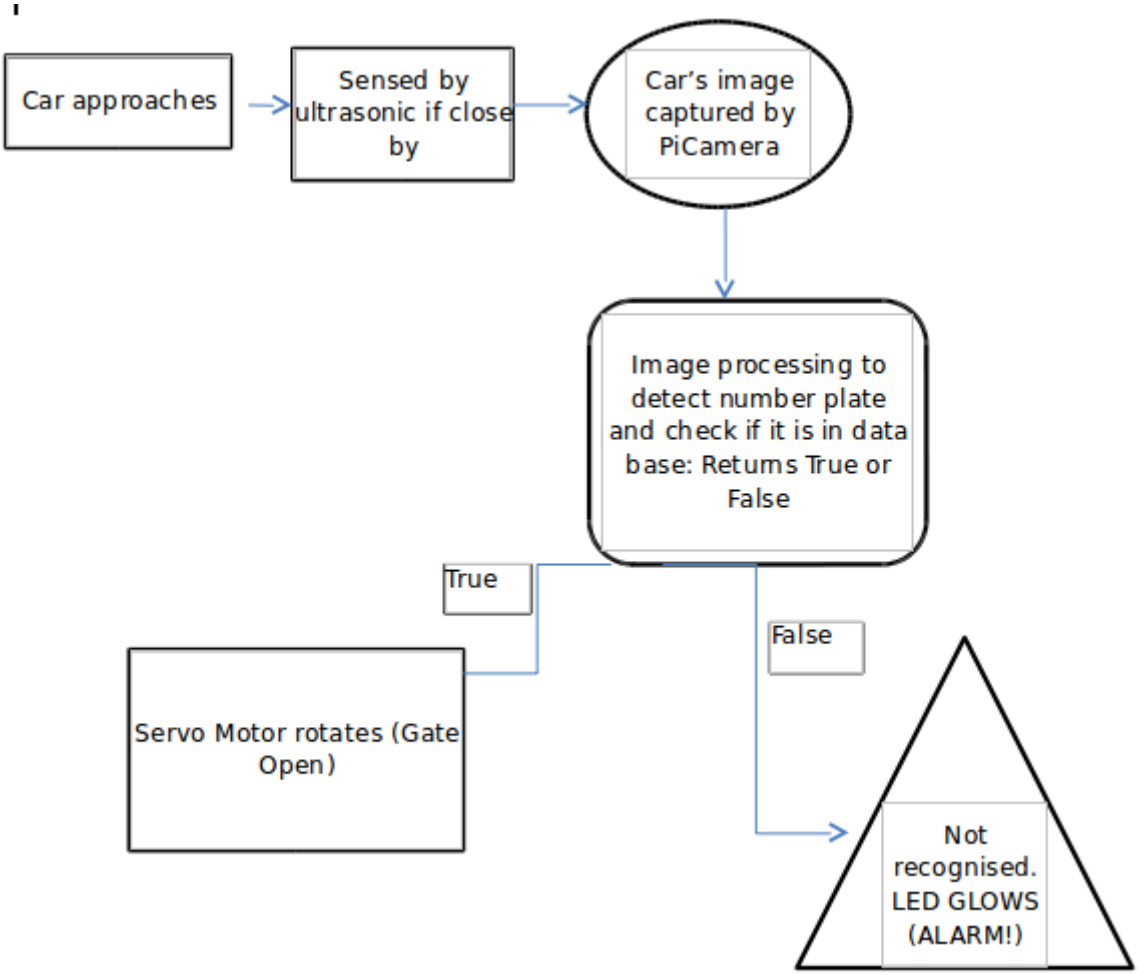
- Raspberry Pi3- the microcontroller board of the system.
- Pi camera – for capturing images.
- Ultrasonic Sensor – For detecting the presence of a vehicle.
- Servo Motor – For opening/closing the gate.
- LED – To alert if the car is not from the community.

2.2 Software Tools

- Python programming language
  - Open CV library
  - Matplotlib Library
  - Python Image Library(PIL)

all are opensource tools.

2.3 System Block Diagram



2.3.1 The key modules involved in the system are:

1. Vehicle detection(using ultrasoni sensor)(identify if a car is there in front of the gate)
2. Number plate identification
3. Number plate autorization (check if the number of the car belongs to that appartments)
    1. Indentify the car number-use image processing
        1. idenify number plate in the image.
        2. get the number plate image as a text.
    2. Search if the number is present in the database
4. Actuation- open gate if car belongs to the community, else buzz an alarm.

2.3.2 The above modules would be interfaced and deployed as follows:

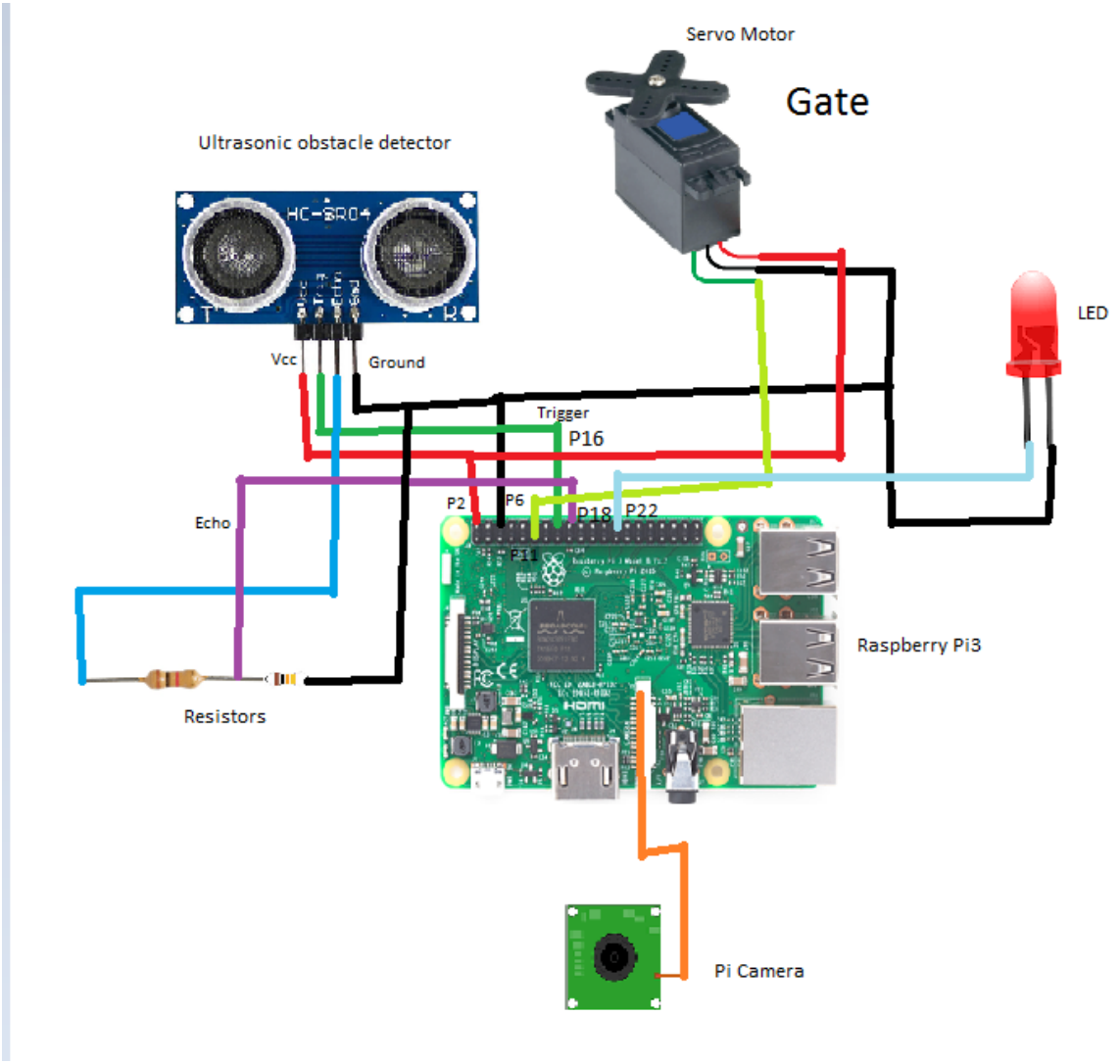
- The first module in the system that initiates the other processes is the the vehicle detection system comprising of an ultrasonic sensor, which considers an object in front of it as a car and informs the camera to capture the picture, provided the car is at a distance of 4m or less. If this module is not there, then the system should be continiuosly be capturing video and must be running image processing algorithms for object(car) detection, which are often slow and resource(power and computation) intensive, moreover the system couln't do any other job while running these algorithms and it may happen that the system may be detecting a car in a frame where a car is not actually present, while a car enters, incresing the wait time of the car.

- A Camera, which is positioned in an appropriate orientation, so that the number plate of any sized car is always capured when at a distance of 0.5-4m from it. Camera is always active, but it captures the image only when the ultasonic sensor signals the camera, indicating presence of a car, which wants to go in.

- The image captured by the camera is sent to a backend python code which segments the number plate part and tries to extract the text on the number plate and check if it is present in the apartment car database. This module is separated from the image capturing module, because it logical to keep hardware(camera) and software(python code) separately as in an embedded system, the software module has a data(image) dependency on the hardware part. The output of this software module is a binary signal- True(if car belongs to the apartments) or False (if it doesn't).

- Based on the output of software module, if it is true then the gate is opened else the gate remains closed and an alert is generated, since we have built a prototype, we make a LED glow to show there is an alert.

2.4 Circuit Diagram



2.4.1 Description

The logical flow of the system has been explained in circuit diagram, now the significance of the components involved is explained here.

2.4.1.1 Ultrasonic sensor

This is used to detect the presence of any object that's close to it. The sensor emits a trigger pulse and anticipates a reflected pulse which gets reflected because of the object. If the time taken to receive the obstacle is short, it means it's close by, and as a result alerts the Pi Camera to take a picture.

2.4.1.2 Pi Camera

This 5MP camera module takes a picture of the object in front of the ultrasonic sensor, if it's close by. The camera then gives the captured image to the raspberry pi for processing and detection of number plate.

2.4.1.3 Raspberry Pi 3

It's a pocket sized computer that's in charge of processing the image to detect the number plate. The tool used for processing is OpenCV, a free python library that can be downloaded. The system searches for a number plate in the image and checks if it's there in an already stored database.

2.4.1.4 Servo Motor

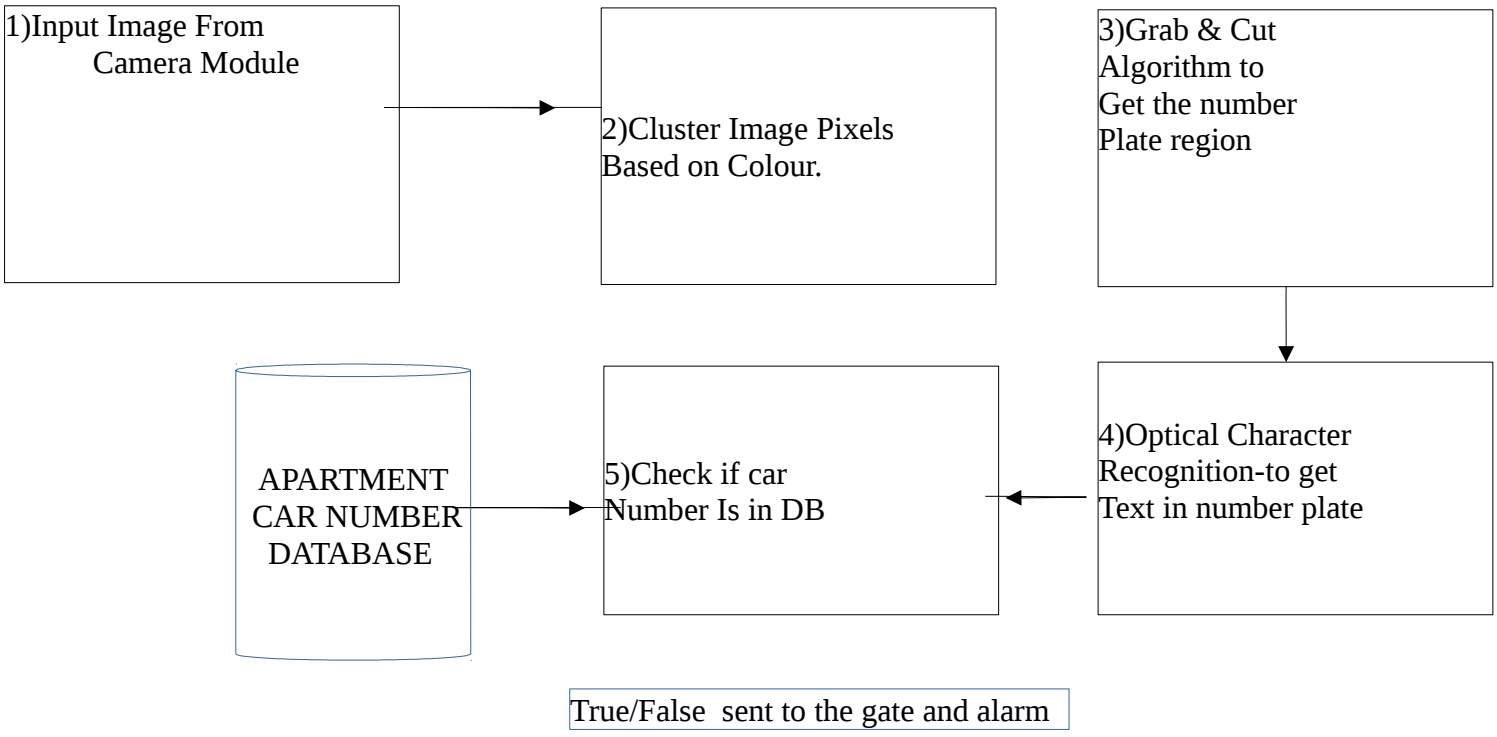
The servo motor is a programmable motor that can rotate based on the width of the pulse given to it from the GPIO pins of the RPi. The motor is used to open a gate, if the captured image's number plate exists in the data base.

2.4.1.5 LED

The led is made to glow if the recognised number plate is not in the data base. As a result, the gate won't open (servo motor won't rotate).

2.5 Software Block Diagram

In section 2.3 we have seen that a picture is captured by the camera sensor and sent to the software module, where image processing is done to get the image, the following is the block diagram of the software module.



2.5.1 Description

A k means clustering algorithm is used to cluster pixels in the image at step1, this is chosen because, number plates either white or yellow in colour with black characters, and hence a region(set of pixels) of the image are of same color therefore we cluster the image by color. In step2, we use an algorithm called grab and cut algorithm,which is an interactive foreground extraction algorithm, where we specify the size of a rectangle, based on the problem domain, it varies. This algorithm considers the pixels in the rectangular box as foreground then algorithm segments it iteratively to get the best result. Its a Gausian Mixture Model(GMM) based algorithm.

The gram and cut algorithm outputs a sgmented region of input image, containing the number plate, this used in step3 by OCR algorithms to detect characters in the image and finally give the number plate text as output. Then in step4 the database is searched if the car number is there in the database and a true/false answer to this question is sent to the next module, in the block diagram seen in section 2.3.

3.0 IMPLEMENTATION

3.1 Code

```
import numpy as np
import cv2
from PIL import Image
import pytesseract
import matplotlib.pyplot as plt
import RPi.GPIO as GPIO #importing PI's GPIO
from time import sleep
from time import time
import picamera #importing library for PI camera
cam = picamera.PiCamera() #instantiating the camera object- used to capture
GPIO.setmode(GPIO.BOARD) ## Initializing the GPIO of PI
GPIO.setwarnings(False) #swithing off the warnings
T = 16    #For the ultasonic transmit(output), pin T=16 will be used.
E = 18    #For the ultasonic Echo(input), pin E=13 will be used.
Sm = 3    #For the Servo motor(output-to send signals to servomotor), pin Sm=3 will be used.
print('Distance calc in progress...')
```

```
GPIO.setup(T, GPIO.OUT)          #setting pint T=16 as output pin
GPIO.setup(E, GPIO.IN)           #setting pint E=13 as input pin
GPIO.setup(Sm, GPIO.OUT)         #setting pint Sm= 3sas output pin
pwm=GPIO.PWM(Sm, 50)             #Setting servo motor's Pulsw Width Modulation to
50%, i.e signal will be high only 50% of a time period
pwm.start(0)                     #starting PWM
cam.start_preview()              #starting camera
def SetAngle(angle):             #function to rotate the servomotor, by given angle
        duty = (angle / 18) + 2
        GPIO.output(Sm, True)
        pwm.ChangeDutyCycle(duty)
        sleep(1)
        GPIO.output(Sm, False)
        print("Hi2")
        pwm.ChangeDutyCycle(0)
while True:                      #the useful part of code begins here and is in infinite loop- since
the Ultrasonic sensor must continiously check for a car.
  GPIO.output(T, False)  #to be careful, ultrasonic transmit is set to low
  print('Sensor is settling...')
  sleep(1)          #waiting for 1s for sensor to settle
  GPIO.output(T, True)      #transmit the signal
  sleep(0.00001)            #wait for 1micro second
  GPIO.output(T, False)     #stop transmission

  while GPIO.input(E) ==0: #checking if signal is received
    s = time()
  while GPIO.input(E) == 1:
    e = time()
  dur = e – s #calculating the duration for travelling and comming back
  d = dur * 17150 #distance = time/2(because time includes for going and comming back) *
speed of sound in cm/s
  d = round(d, 2) #rounding off distance to 2 decimal places

      if d<=400: #when car is at a distance of 4m or less
```

```
sleep(1)
cam.capture('4_1.png')          #capture the picture
                %matplotlib inline  #image processing starts here
                #function to cluster pixels, with default number of clusters=2
                def cluster(fn,K=2):
                    img = cv2.imread(fn)
                    #kernel = np.ones((3,3),np.float32)/9
                    #img = cv2.filter2D(img,-1,kernel) #median filtering
                    Z = img.reshape((-1,3))

                    Z = np.float32(Z) # convert to np.float32

                    criteria = (cv2.TERM_CRITERIA_EPS +
cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0) # define criteria, number of clusters(K) and
apply kmeans()

ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)

                    # Now convert back into uint8, and make original image
                    center = np.uint8(center)
                    res = center[label.flatten()]
                    res2 = res.reshape((img.shape))
                    cv2.imwrite("4_seg1.png",res2) #saving the segmented image by some
name
                    plt.imshow(res2)

                #grab and cut algorithm
                def grabcut(fn):
                    img = cv2.imread(fn)
                    mask = np.zeros(img.shape[:2],np.uint8) #creating a mask, of the order
height*width of image filled with zeros
                    bgdModel = np.zeros((1,65),np.float64)  #initializing the bacground and
foreground areas with 0s
                    fgdModel = np.zeros((1,65),np.float64)
```

#i have observed that the shape of the vertical position of the number plate was always somewhere in the vertically middle part of the image, so taking one of the intial coordinate as img.shape[0]/2 and this parmeter horizontally was almost at the begining but cutting of some pixels would always save some time, so i have put img.shape[1]/8 as the other coordinate, we are giving the rectangle for the grab and cut algorithm, by specifiing the top left corner of rectangle to be (img.shape[0]/2, img.shape[1]/8) and the width of rectangle is img.shape[1]/2+50, its a relatively larger value as number plates are horizontally long the following is the combination of values that worked out

```
                rect =
(int(img.shape[0]/2),int(img.shape[1]/8),int(img.shape[1]/2)+50,500)
                #for small images- if width is <500
                if(img.shape[1]<500):
                        rect = (50,50,500,500)
        #invoking the grab cut inbuilt function in opencv

cv2.grabCut(img,mask,rect,bgdModel,fgdModel,5,cv2.GC_INIT_WITH_RECT)
                mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')
                img = img*mask2[:,:,np.newaxis]
                #writing and reading to avoid datatype errors
                cv2.imwrite("number_plate.png",img)
                img = cv2.imread("number_plate.png",0)
                #thresholding the image
                ret,img1 = cv2.threshold(img,20,255,cv2.THRESH_OTSU)
                #dilating the image
                kernel2 = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
                #saving the image
                cv2.imwrite("number_plate.png",img1)
                plt.imshow(255-img1),plt.colorbar(),plt.show()
        fn = '4_seg1.png'

        cluster('4_1.png')  #invoking the function
        grabcut(fn) #invoking the function

        #using inbuilt function for OCR
```

```
                image_file3 = 'number_plate.png'
                im3 = Image.open(image_file3)
                text = pytesseract.image_to_string(im3)
                print(text.split())

                from difflib import SequenceMatcher
                #function to find the match % for two number plates
                def similar(a, b):
                    return SequenceMatcher(None, a, b).ratio()

                #taking the database to be a text file and reading its content and storing to a list called content
                with open("carDB.txt") as f:
                    content = f.readlines()
                content = [x.strip() for x in content]

                best = ("",50) #intializing best match to a tuple of a string(to store the car number) and the percentage of match
                #taking minimum match percentage to be 50%
                for number in content:
                    match = similar(''.join(text.split()),''.join(number.split()))*100  #splitting and joining strings to remove space in the strings
                    if match>best[1]:  #checking if a number is better mathcing than the best sofar
                            best = (number,match)
                print(best)
                if best[0]!="": #if there exisits a best match number plate, open the gate
                    pwm.start(0)      #start the servo cycle
                    SetAngle(90)     #Servo is told to rotate by 90degrees(so does the gate)
                    pwm.stop()


    else:                       #if object is not<=400cm then try again
```

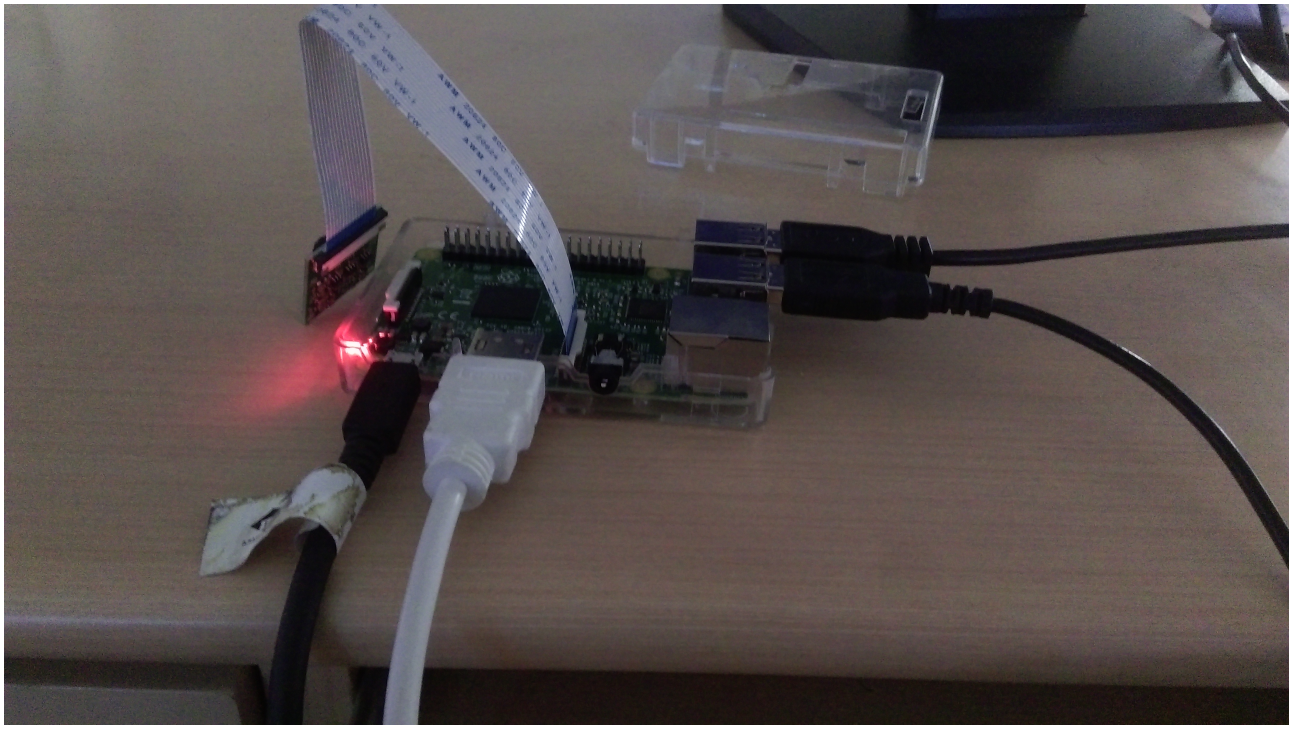
```
        continue
cam.stop_preview()          #stop camera while exiting
```

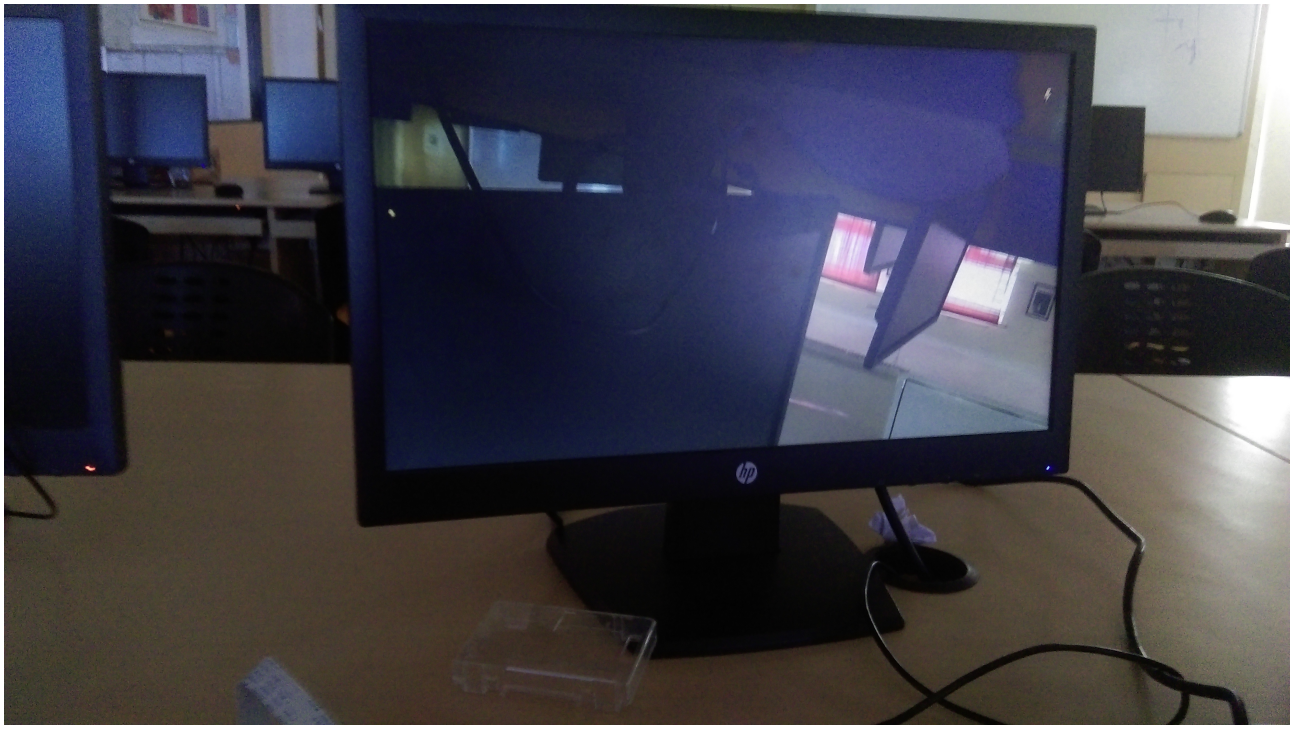
3.2 Hardware

The above code was loaded into Raspberry pi3 and the other hardware components were also connected and were run.

4.0 RESULTS and DISCUSSION

The hardware was setup and code was loaded in to PI and was executed, a sample picture of the hardware set up is like:





A sample run on the image**, acquired from my apartment locality using PI camera**:

Gave the following intermediate results:

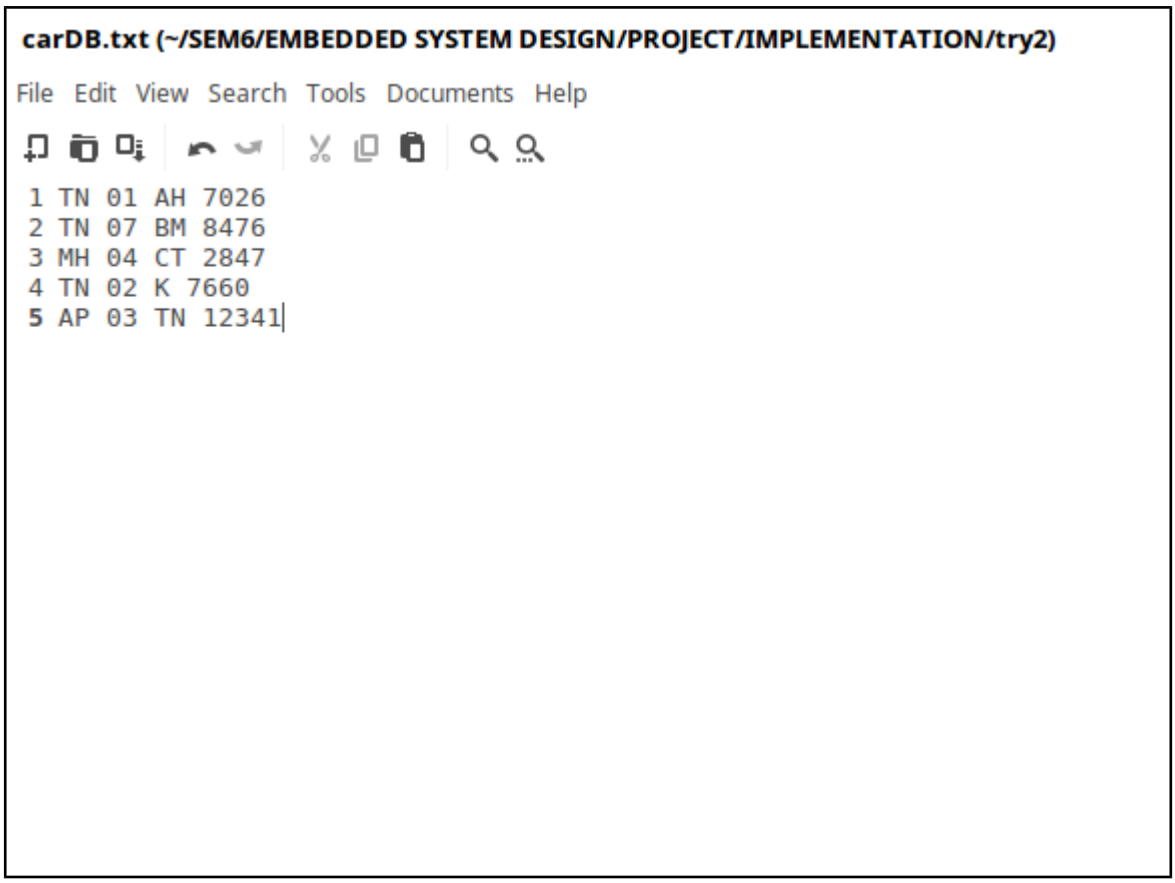
After Clustering:

After Grab and Cut on clustered image:

Text extracted:

```
'TN', '02', 'K7656'
```

A snapshot of the database



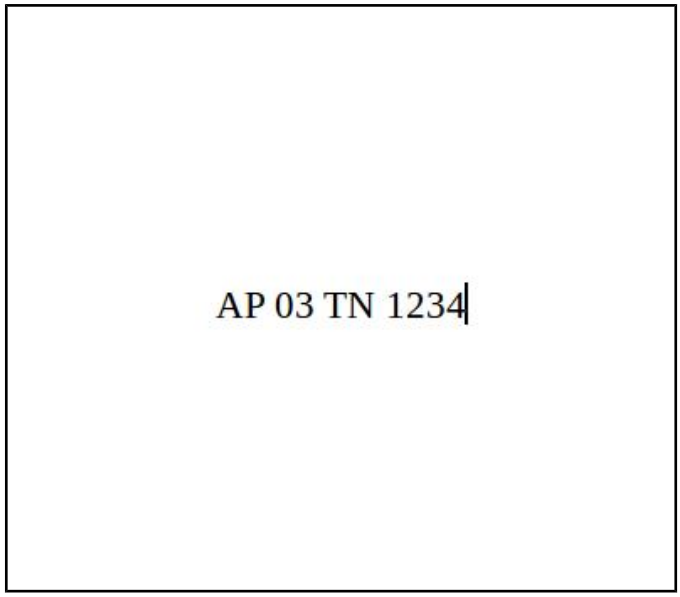
All these OCR algorithms available are not that perfect so, exact matching is not feasible, so we do “soft matching”, with all the images in the databsae , here some amount of error is tolerable, but a minimum match percentage of 50% is set, we get the following best match pair:

```
('TN 02 K 7660', 88.88888888888889)
```

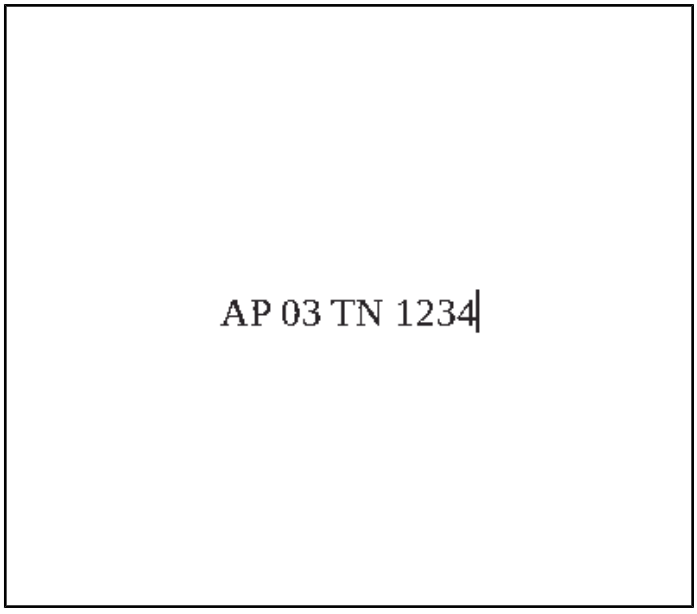
89% matching is good and its not likely that in an apartment community all cars have almost same number plate, so its a safe assumption that if there is>50% match let the car in.

This is one area where there is scope for improvement- the OCR part so that accurate and exact matching could be done.

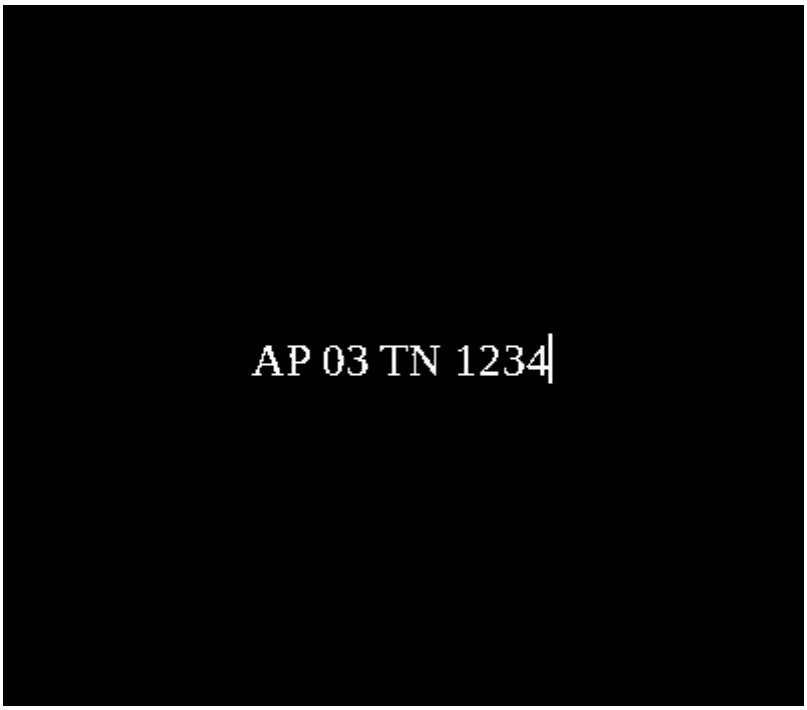
But if we take a synthetic number plate(generated artificially), we get perfect match:



After clustering:



After Grab and Cut:



Text extracted:

```
'AP', '03', 'TN', '12341'
```

Best match : `('AP 03 TN 12341', 100.0)`
its a perfect match!

5.0 CONCLUSION

We have faced a lot of challenges while building working on this project, the primary one being number plate location, there were many research articles, which have first applied median filtering on the images, followed by histogram equalization, then edge detection using sobel and the directly jumped and concluded that the number plate is located, but they were all not working as some noisy area always remained which we couldn't eliminate, it took ~25hrs of effort and reading to come up with a way to locate the plate, after thinking a lot it striked me that a number plate would have pixels of same color therefore clustring similar coloured pixels would help remove unwanted noise and hence upon applying grab and cut algorithm we were able to segment the number plate part, without using any computationally intensive method.

The other challenges were that the ultrosonic sensor was giving random values like 180.36, 15.91,etc even when nothing in front of it moved, after discussing with the lab technecians, we understood that it was a problem with PI and after changing it i got correct values.

We conclude that a prototype of aparment car entry system was designed and implemented as an embedded system.

6.0 REFERENCES

[1] https://opencv.org/
[2] http://www.numpy.org/
[3] http://education.rec.ri.cmu.edu/content/electronics/boe/ultrasonic_sensor/1.html
[4] https://picamera.readthedocs.io/en/release-1.13/
[5] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[6] https://www.edgefx.in/servo-motor-types-and-working-principle/
[7] Vehicle Registration Plate Recognition System Using Template Matching(research paper)

****