EXPERIMENT 8: MINI PROJECT

SELF DRIVING VEHICLE USING IMAGE PROCESSING

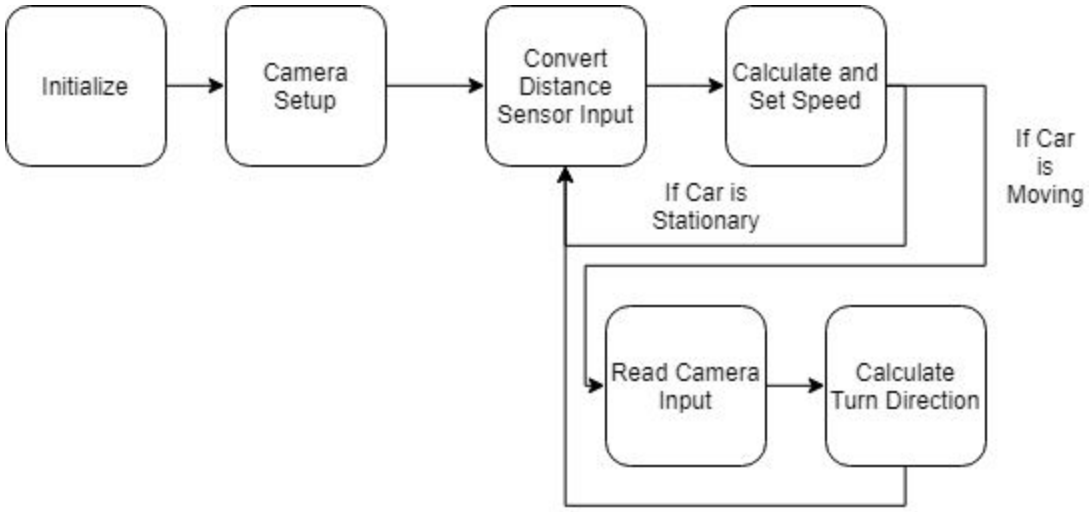
TEAM:

Name	Roll No
Saurabh Parulekar	40
Yash Patil	41
Sopan Phaltankar	45

PROBLEM STATEMENT: A car is moving on a road having traffic. Create a simulation of the vehicles “seen” by the car, process the captured image and show at what speed the car should go, also provide with the details whether the car should change its lane of navigation or not.

SOFTWARE USED : Python with Libraries OpenCV, NumPy

BLOCK DIAGRAM:



PROJECT LINK: https://github.com/yashpatil1998/PCS-Mini-Project

CODE :

```
import cv2              #An Image Processing Library
import numpy as np      #A Library used to store arrays of images
import time             #A Library for time related operations

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
#set the colour parameters here(RGB),"l" is the lower boundary and "h" is the upper boundary
l=[0,0,250]
h=[0,0,255]
lower=np.array(l)#converting python lists to numpy arrays
upper=np.array(h)#converting python lists to numpy arrays

present_speed=0#declaring initial speed

constant=1000000     #P Controller
```

```
average_area=0.0#initial car speed

a=2

choice=input("Enter the Index number to select your selection\n1)Pass an Image\n2)Start Video Stream\nEnter choice:")



def var_speed(n):#Function using Proportional-Controller of PID for calculating speed
    global present_speed
    global a
    if n==0:
        present_speed=5
        return
    present_speed=constant/n
    if(present_speed>=180):
        present_speed=180
        cv2.putText(img,"Speed Limit Reached",(c-200,50),font,0.5,(0,0,255),2)
    print "presentspeed:",present_speed
    return




cx=[]#X co-ordinate of centroid array
cy=[]#Y co-ordinate of centroid array
font = cv2.FONT_HERSHEY_SIMPLEX
if(choice==2):
    cap=cv2.VideoCapture(0)
while(1):
    if(choice==1):
        img=cv2.imread("centerCar.png")
    if(choice==2):
        ret,img=cap.read()
    if(choice==2):
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for (x,y,w,h) in faces:
                cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
                cx.append(x+w/2)
                cy.append(y+h/2)
```

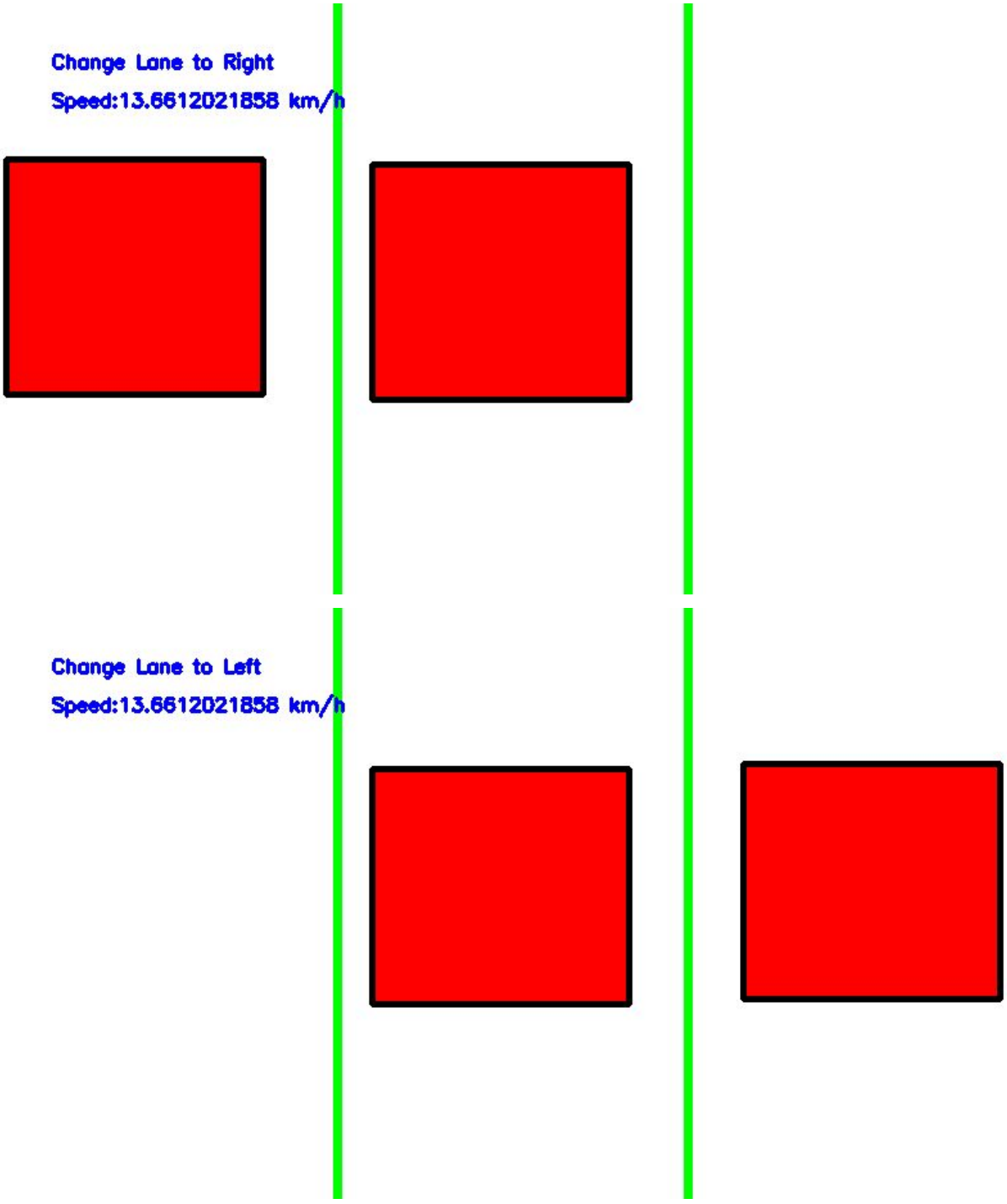
```
                average_area=float(average_area+(float(h)*float(w)))
                cv2.circle(img,(x+w/2,y+h/2),2,(0,255,0),5)


    r,c,ch=img.shape
    #cv2.putText(img,"Hello",(10,r-10),font,0.5,(0,0,255),2)
    mask1=cv2.inRange(img,lower,upper)
    cv2.imshow("mask",mask1)
    _,contours,heirarchy = cv2.findContours(mask1, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    cv2.line(img,(c/3,0),(c/3,r),(0,255,0),5)
    cv2.line(img,(c-c/3,0),(c-c/3,r),(0,255,0),5)
    if(choice==2):
        contours=faces
    #print "contours",len(contours)
    #print "faces",len(faces)
    #print "choice",choice
    if(choice==1):
        for i in contours:
            x,y,w,h = cv2.boundingRect(i)
            average_area=average_area+cv2.contourArea(i)
            cv2.drawContours(img,contours,-1,(0,0,0),3)
            cx.append(x+w/2)
            cy.append(y+h/2)
    if(len(contours)!=0):

            if(choice==1):
                avreage_area=float(average_area)/float((len(contours)))
            print "Average",average_area
            if(len(contours)==0):
                cv2.putText(img,"No Car",(50,50),font,0.5,(255,0,0),2)
            if(len(contours)==1):
                if(cx[0]>(c/3) and cx[0]<(c-c/3)):
                    cv2.putText(img,"Change Lane Either Left or Right",(50,50),font,0.5,(255,0,0),2)
                    var_speed(average_area)
                else:
                    cv2.putText(img,"Straight",(50,50),font,0.5,(255,0,0),2)
                    var_speed(average_area)
            elif(len(contours)==2):
                if((cx[0]<(c/3) and cx[1]>(c-c/3)) or (cx[1]<(c/3) and cx[0]>(c-c/3))):
                    cv2.putText(img,"Straight",(50,50),font,0.5,(255,0,0),2)
                    var_speed(average_area)
                elif(cx[0]>(c/3) and cx[1]>(c/3)):
```

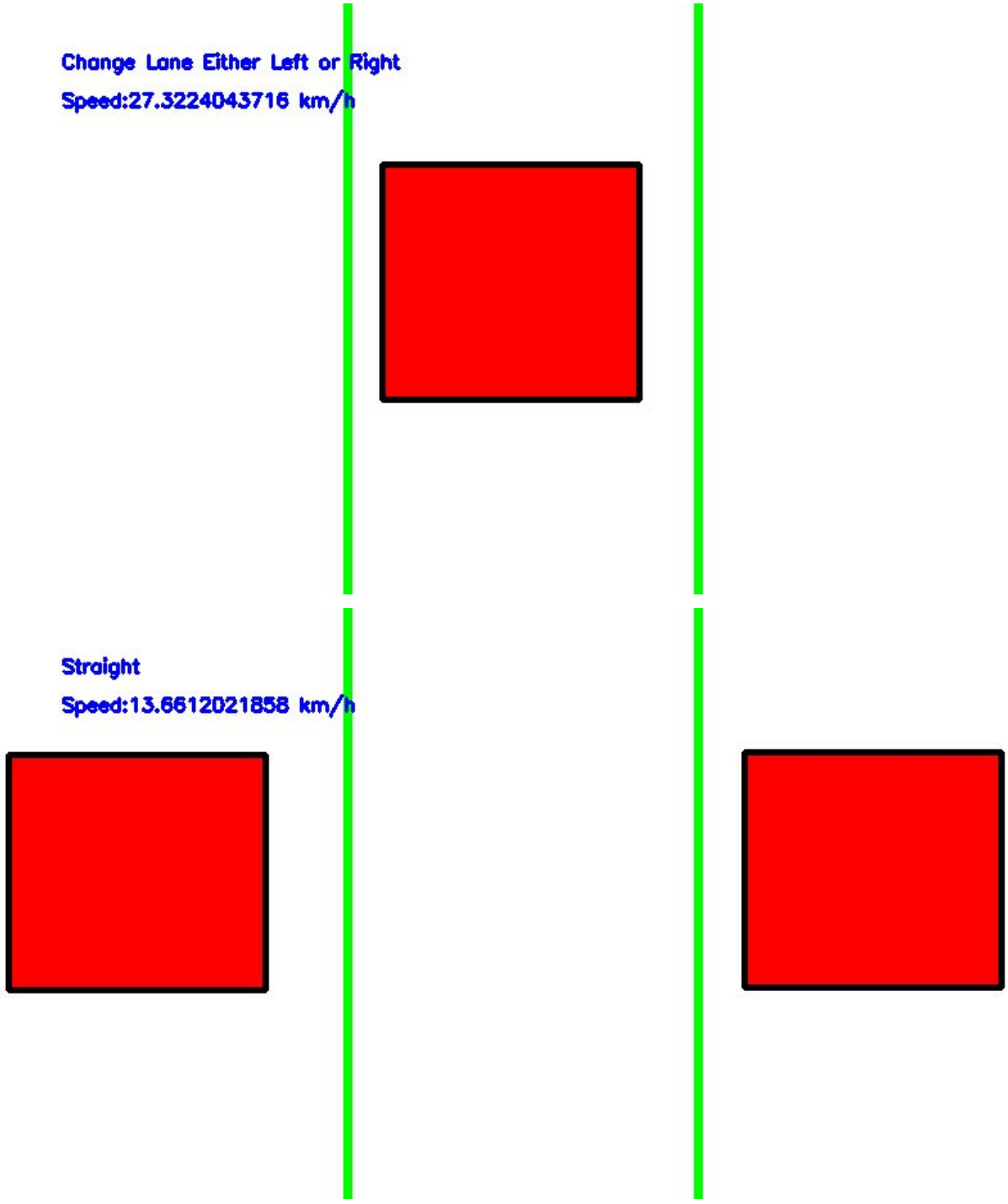
```
                cv2.putText(img,"Change Lane to Left",(50,50),font,0.5,(255,0,0),2)
                var_speed(average_area)
            elif(cx[0]<(c-c/3) and cx[1]<(c-c/3)):
                cv2.putText(img,"Change Lane to Right",(50,50),font,0.5,(255,0,0),2)
                var_speed(average_area)
        else:
            cv2.putText(img,"Constant Speed",(50,50),font,0.5,(255,0,0),2)
            var_speed(average_area)


        cv2.putText(img,"Speed:"+str(present_speed)+" km/h",(50,80),font,0.5,(255,0,0),2)
        #cv2.putText(img,str(present_speed),(110,80),font,0.5,(255,0,0),2)
    cv2.imshow("Car",img)
    average_area=0
    cx=[]
    cy=[]
    k=cv2.waitKey(1)
    if(k==27):
        break

if(choice==2):cap.release()
cv2.destroyAllWindows()
```

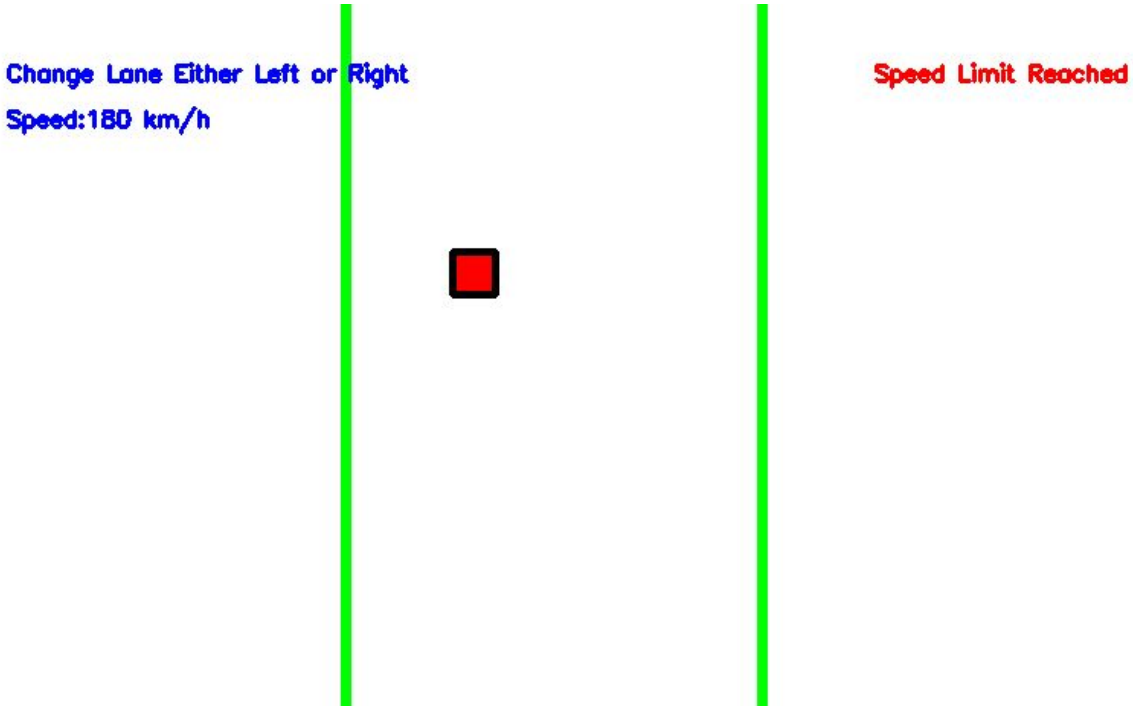
OUTPUT:

The following outputs are for case 1 (pass an image).

Assuming that the red box is the vehicle in front of our car, we have created some of the possible test cases that can be seen practically.


Change Lane to Right
Speed:13.6612021858 km/h
Change Lane to Left
Speed:13.6612021858 km/h


Change Lane Either Left or Right
Speed:27.3224043716 km/h
Straight
Speed:13.6612021858 km/h



CONCLUSION :

- In this experiment we used OpenCV library of Python to do our operations on the captured image.
- We created a software simulation showing the view which a car should see the traffic in front of it. It forms a closed loop control system, the feedback being the area of the vehicle in front of out car (which is proportional to the distance between the vehicle in front of out car and our car).
- The car speed is hence inversely proportional to this feedback. If the distance is large, the speed of the car is increased. If the distance is less, the car speed is reduced and it is also notified to change the lane.