SMATRACON
(Smart Traffic Control Using Image Processing)

A Term Paper Report

Submitted in partial fulfilment of the requirements for

the award of the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

G. SATYA SIVA RAMADAS (150030288)

S. YASWANTH SAI RAM (150030865)

V.MANI VENKATA SAI (150070405)

Under the supervision of

Mr.K. SRIPATHI ROY

Asst. Professor

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

(DST-FIST Sponsored Department)

K L E F

Green Fields, Vaddeswaram, Guntur District-522 502

2017-2018

K L E F

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that this term paper report entitled **“SMATRACON - Smart Traffic Control Using Image Processing”** is a bonafide work done by “**G. Satya Siva Ramadas (150030288) S. Yaswanth Sai Ram (150030865) V. Mani Venkata Sai (150070405)”** in partial fulfilment of the requirement for the award of degree in **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE ENGINEERING** during the academic year 2017-2018.

Mr. K. SRIPATHI ROY | **Dr. V. HARI KIRAN**

(Term Paper Guide) | **(HOD CSE)**

K L E F

DEPARTMENT OFCOMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that this term paper report entitled **“SMATRACON - Smart Traffic Control Using Image Processing”** has been prepared by us in partial fulfilment of the requirement for the award of degree “**BACHELOR OF COMPUTER SCIENCE AND ENGINEERING**” during the academic year 2017-2018.

We also declare that this term paper report is of our own effort and it has not been submitted to any other university for the award of any degree.

Date: 26-04-2018

Place: Vaddeswaram.

Name of the Student	**Univ. Reg. No**
G. SATYA SIVA RAMADAS	**150030288**
S. YASWANTH SAI RAM	**150030865**
V.MANI VENKATA SAI	**150070405**

ACKNOWLEDGEMENT

Our sincere thanks to **Mr. K. SRIPATHI ROY,** Asst. Professor for the outstanding support throughout the research for the successful completion of the work

We express our gratitude to **Dr. V. HARI KIRAN,** Head of the Department of Computer Science and Engineering for providing us with adequate facilities, ways and means by which we can complete this term paper work.

We would like to place on record the deep sense of gratitude to the honourable Vice Chancellor, KLEF for providing the necessary facilities to carry the concluded term paper work.

Last but not the least, we thank all Teaching and Non-Teaching Staff of our department and especially my classmates and my friends for their support in the completion of our term paper work.

Table of Contents

CONTENTS		PG.NO
1.	ABSTRACT	6
2.	INTRODUCTION	7
3.	LITERARY SURVEY	8
4.	OBJECTIVES INCLUDED IN THE PROJECT	11
5.	BLOCK DIAGRAM	11
6.	PROBLEM STATEMENT	12
7.	IMPLEMENTATION	12
8.	PROGRAM CODE & OUTPUT	19-26
9.	CONCLUSION & FUTURE SCOPE	27
10.	REFERENCES	27

ABSTRACT:

Now-a-days traffic management has become one of the most severe problems in the world due to the growth of industrialization and population. Due to the tremendous growth in the traffic, control of traffic by humans became difficult and so machine powered traffic controllers were developed using the modern technologies. In smart way, we exploit the new and easy technique called as “SMATRACON – SMART TRAFFIC CONTROL using Image Processing”. In the busy traffic we can easily identify the vehicle image by attached cameras with the embedded technology and then undergo the process with the images of the 2-wheeler vehicle and 4-wheeler vehicle provided and then match them locate and count the number of vehicles thereby moderating of traffic can be done. This method uses Image processing for the future enhancement such that usage of man power can be reduced. Machine power here plays a key role so that traffic moderation can be done. In this we use the technique digital processing so that the count of 2-wheeler and 4-wheeler vehicles can be displayed.

INTRODUCTION:

To solve the traffic problem, there are many ways but a fast, economical and efficient traffic control system for national development and public sustenance is needed. Government decided to impose fines on the persons who violate traffic rules but in vain. So, among all the best ways of improving traffic flow and safety of the current transportation system, practical application of automation and intelligent control methods to roadside infrastructure as well as vehicles is the best and quick method. This method uses Image processing for the future enhancement such that usage of man power can de reduced. Machine power here plays a key role so that traffic moderation can be done. In this we use the technique digital processing so that the count of 2-wheeler and 4-wheeler vehicles can be displayed.

To moderate and examine the traffic in efficient and smart way, we exploit the new and easy technique called as “SMATRACON – SMART TRAFFIC CONTROL using Image Processing”.

SMART TRAFFIC
CONTROL

LITERATURE SURVEY:

Image Processing:

Image processing is a method to perform some operations on an image, to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image.

Image Processing basically needs following three steps:

(1) Importing the image via image acquisition tools.

(2) Analysing and manipulating the range.

(3) Report that is based on image analysis.

Image processing refers to quantitative analyses and/or algorithms applied to digital image data. It allows generation of 3D parametric maps and implies calculation of values that should be ultimately replicable and rather-independent. Generally, 2 types of image processing are used.

They are:

(1) ANALOGUE IMAGE PROCESSING

(2) DIGITAL IMAGE PROCESSING

Analogue Image Processing:

Analog image processing is any image processing task conducted on two-dimensional (2-D) analog signals and if the pictorial representation of the data represented in analog wave formats that can be named as analog image. Mainly, analogue image processing can be used for the hard copies like printouts and photographs.

Analog Signal

Digital Image processing:

Digital image processing deals with manipulation of digital images through a digital computer. It is a subfield of signals and systems but focus particularly on images. The input of that system is a digital image and the system process that image using efficient algorithms and gives an image as an output by using computers. The three general phases that all types of data undergo while using digital technique are pre-processing, enhancement, and display, information extraction.

Digital Signal

OpenCV:

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel’s Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations. OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

OpenCV

OpenCV-Python:

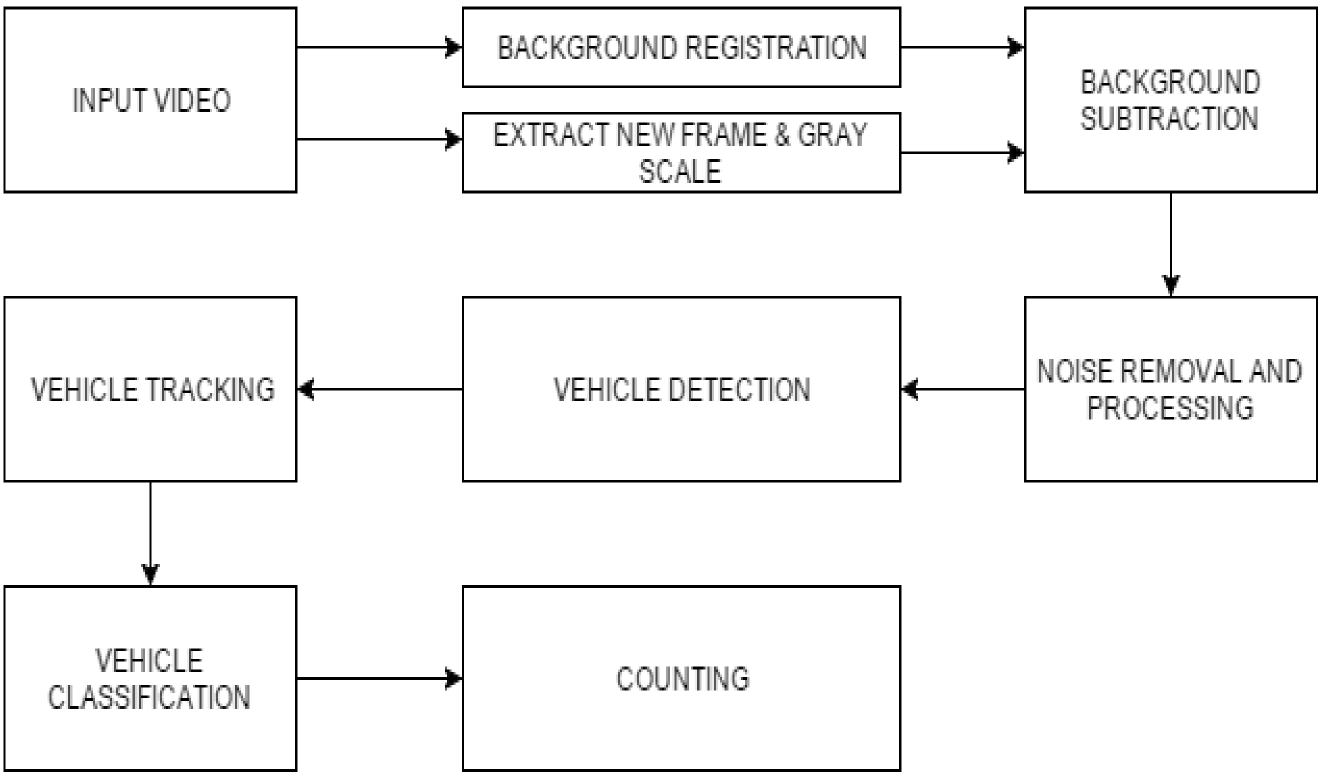
Python is a general-purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. Compared to other languages like C/C++, Python is slower but can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python.

Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this. So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems. OpenCV-Python works with python wrapper around original C++ implementation. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV.

OBJECTIVES INCLUDED IN THE PROJECT:

1) To develop the system which can be efficient using artificial intelligence, such that the concept image processing technique is implemented for estimation of actual road traffic in the particular area by counting the number of vehicles in the traffic up to the particular distance
2) To monitor the system in such a way that the count of vehicles is displayed in digital way
3) Developing the server to handle previous data in automatic way.
4) To develop a vision-based surveillance system capable of identifying vehicles in the scene.
5) To track the vehicles as they progress along the image sequence.
6) To classify the vehicles as 2 wheelers and 4 wheelers
7) To count the number of vehicles passing.

BLOCK DIAGRAM:



PROBLEM STATEMENT:

Due to the heavy population in the cities as well as villages traffic problems like heavy traffic jams, violation of traffic rules, improper traffic management and traffic congestion increased enormously that results in long waiting times, loss of life, wastage of fuel and money, accidents,etc.,. Vehicle detection and counting is very important in traffic congestion, keep tracking of vehicles and to control the traffic signal duration. Enormous growth of population results in the improper management of traffic and thus it results in th lack of control of traffic by human power.As a result fines are imposed on those who viloate traffic rules but became invain in some cases. In order to reduce the traffic problem, we need the smart solution and it can be possible through “SMATRACON-Smart Traffic Control Using Image Processing”

IMPLEMENTATION:

Background Registration:

Background registration is the most important job in vision-based surveillance systems. The background is created by taking the continuous average of accumulated weight of first few frames and is gray scaled.

Background Registration algorithm:

```
1)    procedure get background (alpha , no of frames)
2)    result = empty frame
3)        for 1 : no of frames do
4)            Retrieve new frame from the camera
5)            result = ( 1 - alpha )result + ( alpha )new frame
6)        end for
7)    result = grayscale(result)
8)    return result
9)    end procedure
```

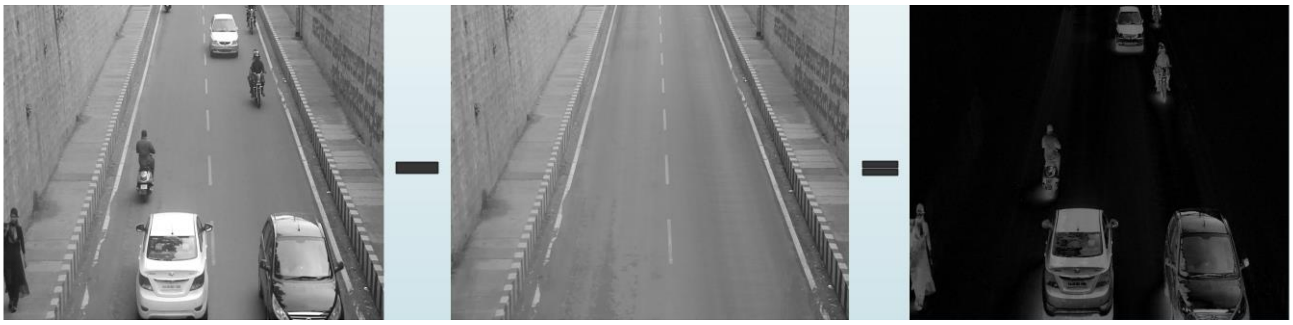
Frame Extraction and Grayscaling:

The video to be processed in nothing but a series of images or frames. Each frame is accessed individually and processed. OpenCV provides us with methods to extract individual frames with ease. Before processing the frame, we convert it to grayscale. Converting image to grayscale is a simple task in OpenCV. It provides a function" cvtColor()" for the same.It uses the following formula.

$$Y = 0.299R + 0.587G + 0.114B$$

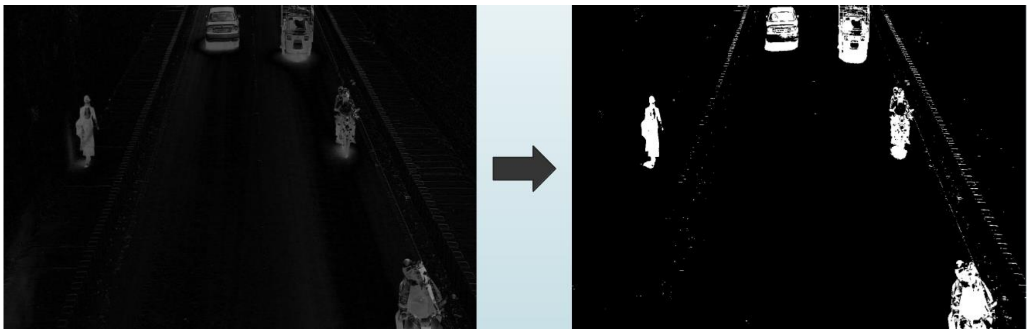
Background Subtraction:

Background subtraction is done to obtain the objects in the fore-ground i.e for detection of vehicles. Here we take the absolute difference between theregistered background and the current frame. Since both the images are grayscaled it isvery easy to calculate the difference and for further processing. OpenCV provides us witha function called "absdiff()" to carry out the process.



Thresholding:

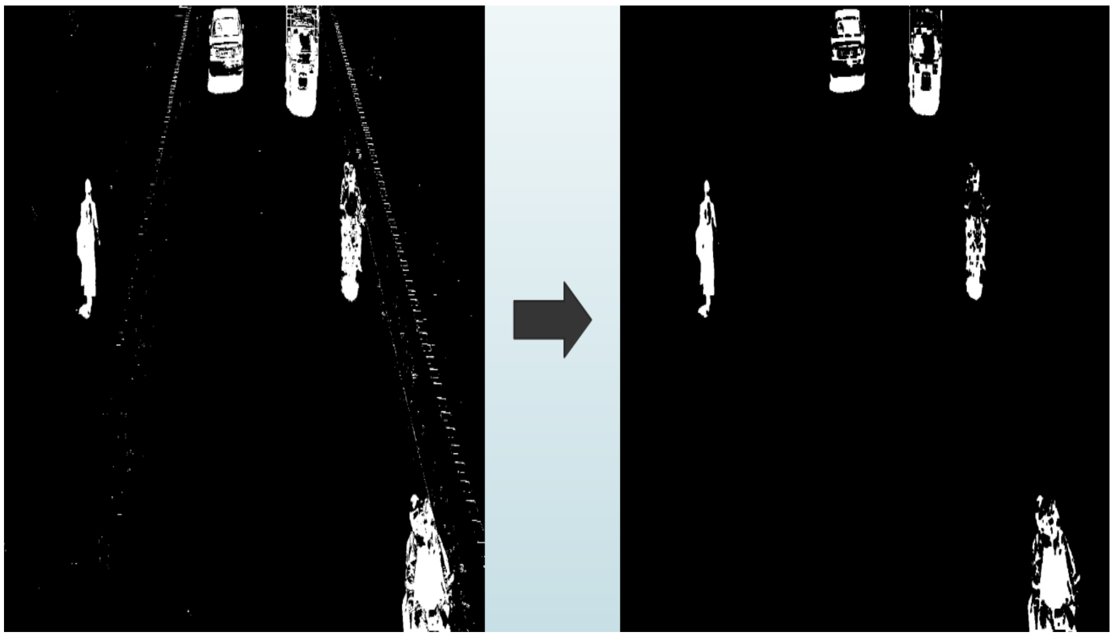
After obtaining the subtracted image,the image has to be segmented into regions containing vehicle and the regions containing a non-vehicle. This is decided on the intensity of the region. If the region has an intensitygreater than a specified threshold, it is converted to white and the rest is black. OpenCV provides a function "threshold()" for this purpose.



Erosion:

Erosion is one of two fundamental operations in morphological image processing from which all other morphological operations are based. The erosion operator takes two pieces of data as inputs. The first is the image which is to be eroded. The second is a (usually small) set of coordinate points known as a structuring element (also known as a kernel). It is this structuring element that determines the precise effect of the erosion on the input image. The mathematical definition of erosion for binary images is as follows:

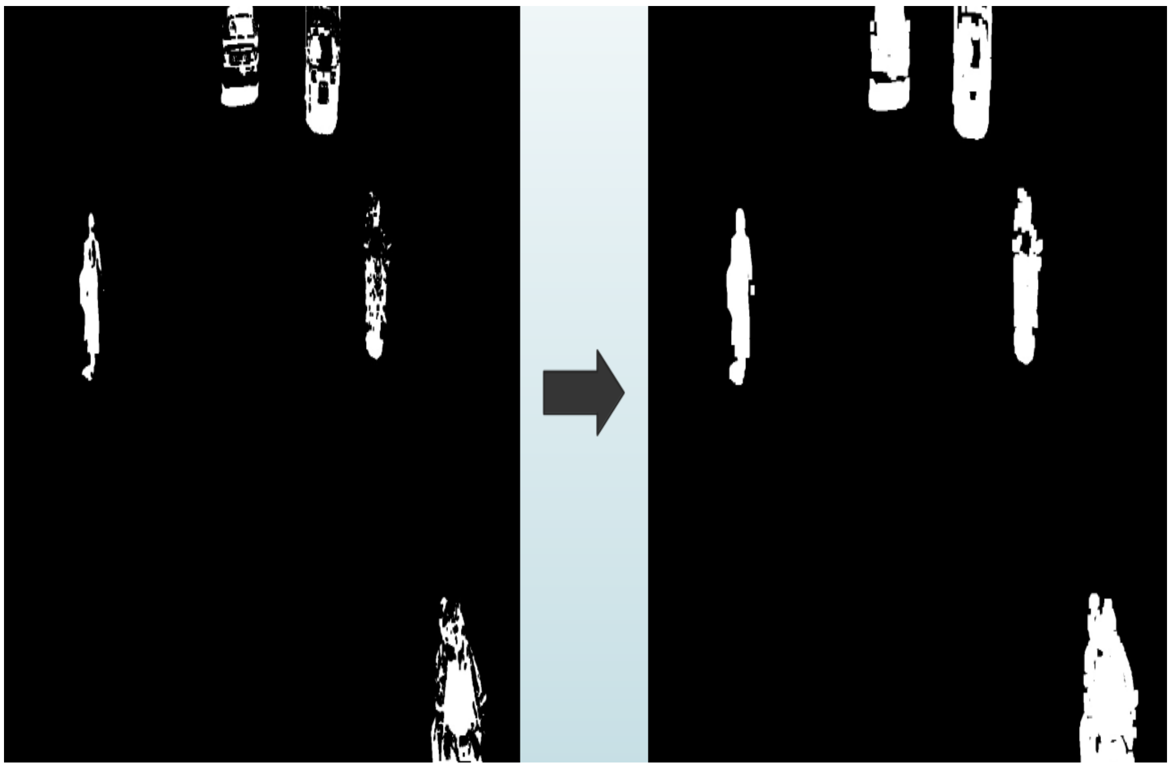
We consider each of the foreground pixels in the input image in turn. For each foreground pixel 10 (which we will call the input pixel) we superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel coordinates. If for every pixel in the structuring element, the corresponding pixel in the image underneath is a foreground pixel, then the input pixel is left as it is. If any of the corresponding pixels in the image are background, however, the input pixel is also set to background value. OpenCV provides "erode()" function for this process.



Dilation:

Dilation is one of the two basic operators in the area of mathematical morphology, the other being erosion.The dilation operator takes two pieces of data as inputs. The first is the image which is to be dilated. The second is a (usually small) set of coordinate points known as a structuring element (also known as a kernel). It is this structuring element that determines the precise effect of the dilation on the input image. To compute the dilation of a binary input image by this structuring element, we consider each of the background pixels in the input image in turn. For each background pixel 11(which we will call the input pixel) we superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel position.

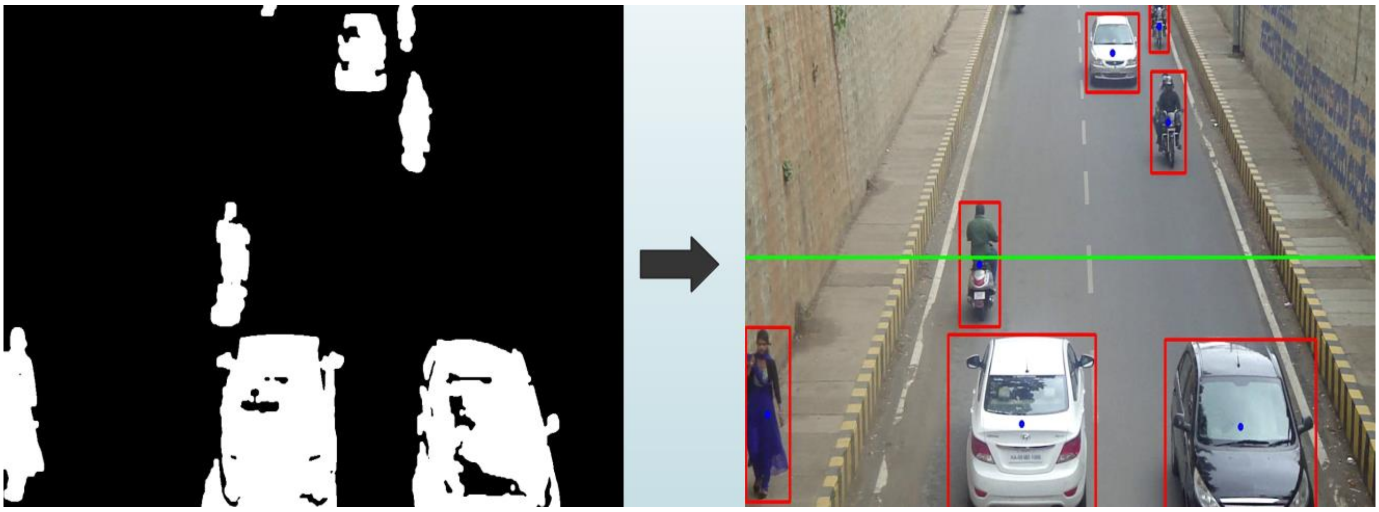
If at least one pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value. If all the corresponding pixels in the image are background, however, the input pixel is left at the background value. OpenCV provides “dilate()” function for this process.



Vehicle Detection:

After the noise removal, the frame is then ready for vehicle detection. All the regions that are white (or 1) are considered as vehicles and the rest as non- vehicles. We take contours of all the vehicle regions, calculate the smallest rectangle that will fit the contour and draw bounding boxes. OpenCV provides the functions "findContours()" which finds the contours for all the white regions in the frame.

Another function "boundingRect()" calculates the coordinates and size of smallest rectangle to fit the contour. These rectangles are drwn on the frame. The following figure shows drawing bounding boxes around the vehicles. After this the vehicles are tracked using the centroids of these bounding boxes.



Vehicle Counting and Classification:

Counting:

We employ a small tricky algorithm for counting.A vehicle is counted only when the centroid of its bounding box passes through the virtual line.To make sure that a vehicle is counted only once , we also store the centroids of previous frame and use them comparison.A vehicle will be counted if its centroid of current frame has crossed the virtual line and the centroid of previous frame has not crossed the line.

Algorithm Check if the vehicle crossed the line:

```
1) procedure Check crossing(count , old centroids , current centroids
2)     for each centroid c1 in current centroids do
3)         c2 = position of centroid in previous frame
4)         if c1 is above the virtual line and c2 is below the virtual line then
5)             count = count + 1
6)         end if
7)         if c1 is below the virtual line and c2 is above the virtual line then
8)             count = count + 1
9)         end if
10)    end for
11) return count
12) end procedure
```

Classification:

As soon as the vehicle crosses the line , the next job is classification of vehicles is an easy task. Here we consider the area under the contours. If the area of the boxes is greater than a specified threshold , it is counted as a 4 wheeler else a 2 wheeler. OpenCV in this regard provides a function “contourArea()” to calculate area of the contours which helps in classification.

Classification of 2- wheeler and 4-wheeler:

2-wheeler:

A person wearing helmet and riding the bike is taken and assumed as the 2-wheeler in this program and then counting is considered.

4-wheeler:

A person riding the car and the vehicle which possess 4 wheels on the road is taken and assumed as the 4-wheeler in this program and then counting is considered.

PROGRAM CODE:

```
import cv2

import numpy as np

import math

import sys


def draw_previous():

    for (x,y,w,h) in old_boxes:

        cv2.circle(frame,(getCentroid(x,y,w,h)),3,(255,0,255),3)


def getCentroid(x,y,w,h):

    return x+int(w/2),y+int(h/2)


def getCentroids(boxes):

    centroid = []

    for (x,y,w,h) in boxes:centroid.append(getCentroid(x,y,w,h))

    return centroid


def distance(point_1,point_2):

    return (math.sqrt( (point_2[0] - point_1[0])**2 + (point_2[1] - point_1[1])**2 ))


def see_distance(old_centroids,new_centroids):

    i = 0

    distances = []

    while i < len(old_centroids) and i < len(new_centroids):

        distances.append(distance(old_centroids[i],new_centroids[i]))

        i+=1

    print(distances)
```

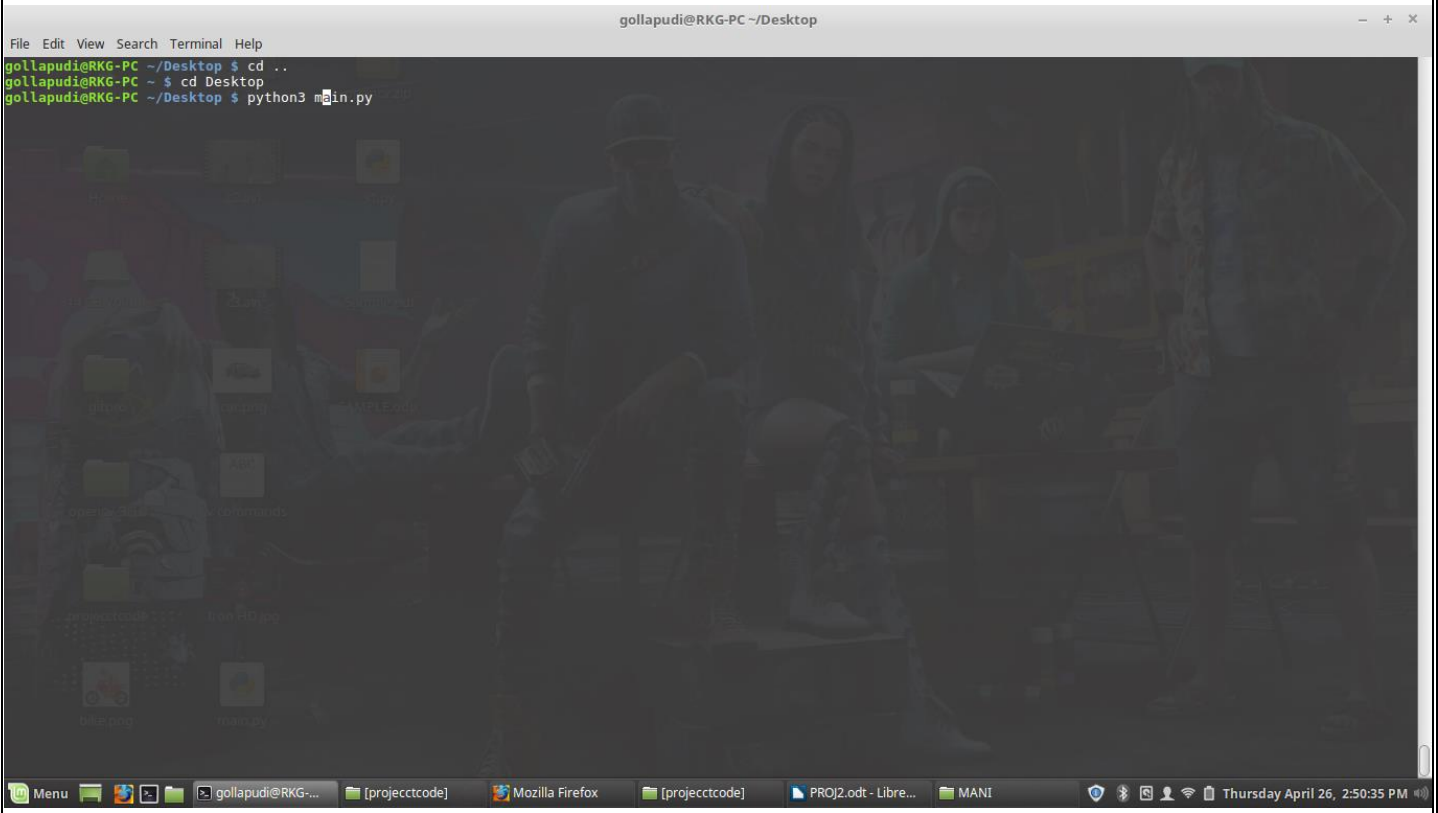
```
def get_closest(p1,new_centroids):
        minimum , index = sys.maxsize , 0
        for i in range(len(new_centroids)):
                d = distance(p1,new_centroids[i])
                if d <= minimum:
                        minimum = d
                        index = i
        # print(index,len(new_centroids))
        if index == 0 and len(new_centroids) == 0:new_centroids.append(p1)
        return index , minimum , new_centroids[index]

# Function to check if any vehicle corsses the line
def check(height,count_cars,count_bikes,old_centroids,new_centroids,contours,threshold,area):
        # print(old_centroids)
        # print(new_centroids)
        for p1 in old_centroids:
                if p1[1] >= height:
                        index , distance , p2 = get_closest(p1,new_centroids)
                        if distance > threshold:continue
                        if p2[1] < height:
                                if cv2.contourArea(contours[index]) > area:
                                        count_cars += 1
                                else:count_bikes += 1
                                print("\r",end="")
                                # print("Count: ",count,end="")
                                continue
                        # print(p1,p2,distance,height)
        if p1[1] <= height:
            index , distance , p2 = get_closest(p1,new_centroids)
            if distance > threshold:continue
            if p2[1] > height:
                if cv2.contourArea(contours[index]) > area:
                    count_cars += 1
                else:count_bikes += 1
                print("\r",end="")
                # print("Count: ",count,end="")
            # print(p1,p2,distance,height)
    return count_cars , count_bikes



# Function to draw bounding box around contours
def drawBoundingBox(opening,frame,count_cars,count_bikes,min_length,width,height):
    global old_boxes , new_boxes
    _,contours,hierarchy                                                    =
cv2.findContours(opening,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    old_boxes , new_boxes , cnts = new_boxes[:] , [] , []
    for cnt in contours:
        x,y,w,h = cv2.boundingRect(cnt)
        if w < min_length or h < min_length: continue
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),3)
        cx , cy = getCentroid(x,y,w,h)
        cv2.circle(frame,(cx,cy),3,(255,0,0),3)
        new_boxes.append((x,y,w,h))
        cnts.append(cnt)
    cv2.line(frame,(0,height),(width,height),(0,255,0),3)
    old_centroids , new_centroids = getCentroids(old_boxes) , getCentroids(new_boxes)
```

```
	# draw_previous()
	count_cars , count_bikes = check(height,count_cars,count_bikes,old_centroids,new_centroids,cnts,40,12500) # 7000 or 12500
	# print("\n",old_boxes)
	# print(new_boxes,"\n","*"*150,"\n")
	return count_cars , count_bikes , frame


# Function to get the background from frames the by averaging
def getBackground(cam,count,step):
	ret ,frame = cam.read()
	avg = np.float32(frame)
	for i in range(1,count):
		for j in range(step):ret ,frame = cam.read()
		cv2.accumulateWeighted(frame,avg,0.01)
		res = cv2.convertScaleAbs(avg)
		print("\r",end="")
		print("Initialising",int(i/count*100),"%",end="")
	print("\rInitialising 100 %")
	background = cv2.cvtColor(res,cv2.COLOR_BGR2GRAY)
	cv2.imshow("final_frame",frame)
	return background


# Adding the results to the output
def display_final(frame,size,mask,count,x1,y1,x2,y2):
	roi = frame[x1:size[0]+x1,y1:size[1]+y1]
	bg = cv2.bitwise_and(roi,roi,mask=mask)
	frame[x1:size[0]+x1,y1:size[1]+y1] = bg
	font = cv2.FONT_HERSHEY_SIMPLEX
	cv2.putText(frame,str(count),(x2,y2), font, 2,(0,0,0),3)
```

```
    return frame

source = '../videos/c2.avi'
cam = cv2.VideoCapture(source) # Defining the Camera
background = getBackground(cam,300,2) # Getting the background image
cv2.imshow('background',background)

cam = cv2.VideoCapture(source) # Re-Defining the Camera

kernel1 = np.ones((3,3),np.uint8)
kernel2 = np.ones((5,5),np.uint8)
width , height = background.shape[1] , background.shape[0]
new_boxes , old_boxes ,count_cars,count_bikes = [] , [] , 0 , 0

# Car icon to display count at the top
car = cv2.imread('car.png',0)
bike = cv2.imread('bike.png',0)
size_car = car.shape
bike = cv2.resize(bike,(int(bike.shape[0]/2),int(bike.shape[1]/2)))
size_bike = bike.shape
ret , mask_car = cv2.threshold(car,220,255,cv2.THRESH_BINARY)
ret , mask_bike = cv2.threshold(bike,220,255,cv2.THRESH_BINARY)

while(1):
    ret ,frame = cam.read()
    if not ret:break
    # Preprocessing the image
    frame_gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY) # Conversion to gray-scale
```

```
        frame_diff = cv2.absdiff(background,frame_gray) # Background subtraction

        ret , thresh = cv2.threshold(frame_diff,35,255,cv2.THRESH_BINARY) # Thresholding the
subtracted frame

        erosion = cv2.erode(thresh, kernel1, iterations=1) # Filtering by erosion

        dilation = cv2.dilate(erosion,kernel2 , iterations=2) # Filling holes by dilation

        median = cv2.medianBlur(dilation,5) # Median filtering for smoothing


        count_cars, count_bikes , final =
drawBoundingBox(median,frame,count_cars,count_bikes,30,width,int(height/2)) # Drawing
contours

        # Displaying the frames

        # cv2.imshow('median',median)

        # cv2.imshow('thresh',thresh)

        # cv2.imshow('erosion',erosion)

        # cv2.imshow('dilate',dilation)

        final = display_final(frame,size_car,mask_car,count_cars,0,0,220,80)

        final = display_final(final,size_bike,mask_bike,count_bikes,0,350,500,80)

        cv2.namedWindow('final',cv2.WINDOW_NORMAL)

        cv2.resizeWindow('final', 1366,768)

        cv2.imshow('final',final)

        # print(boxes)

        k = cv2.waitKey(30) & 0xFF

        if k == ord('q'):break


print()

cv2.destroyAllWindows()

cam.release()
```

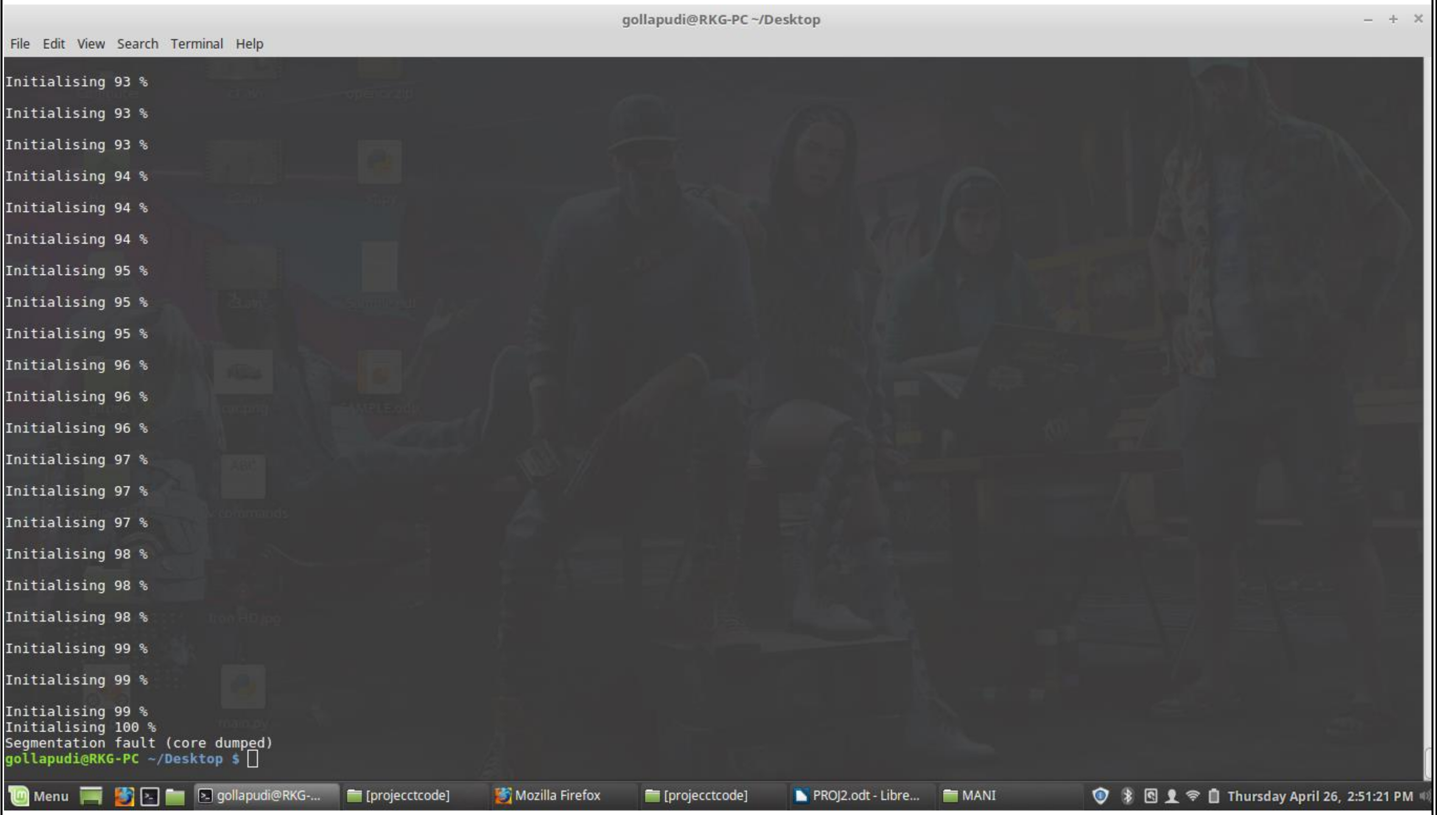
OUTPUTS:



gollapudi@RKG-PC ~/Desktop
File Edit View Search Terminal Help

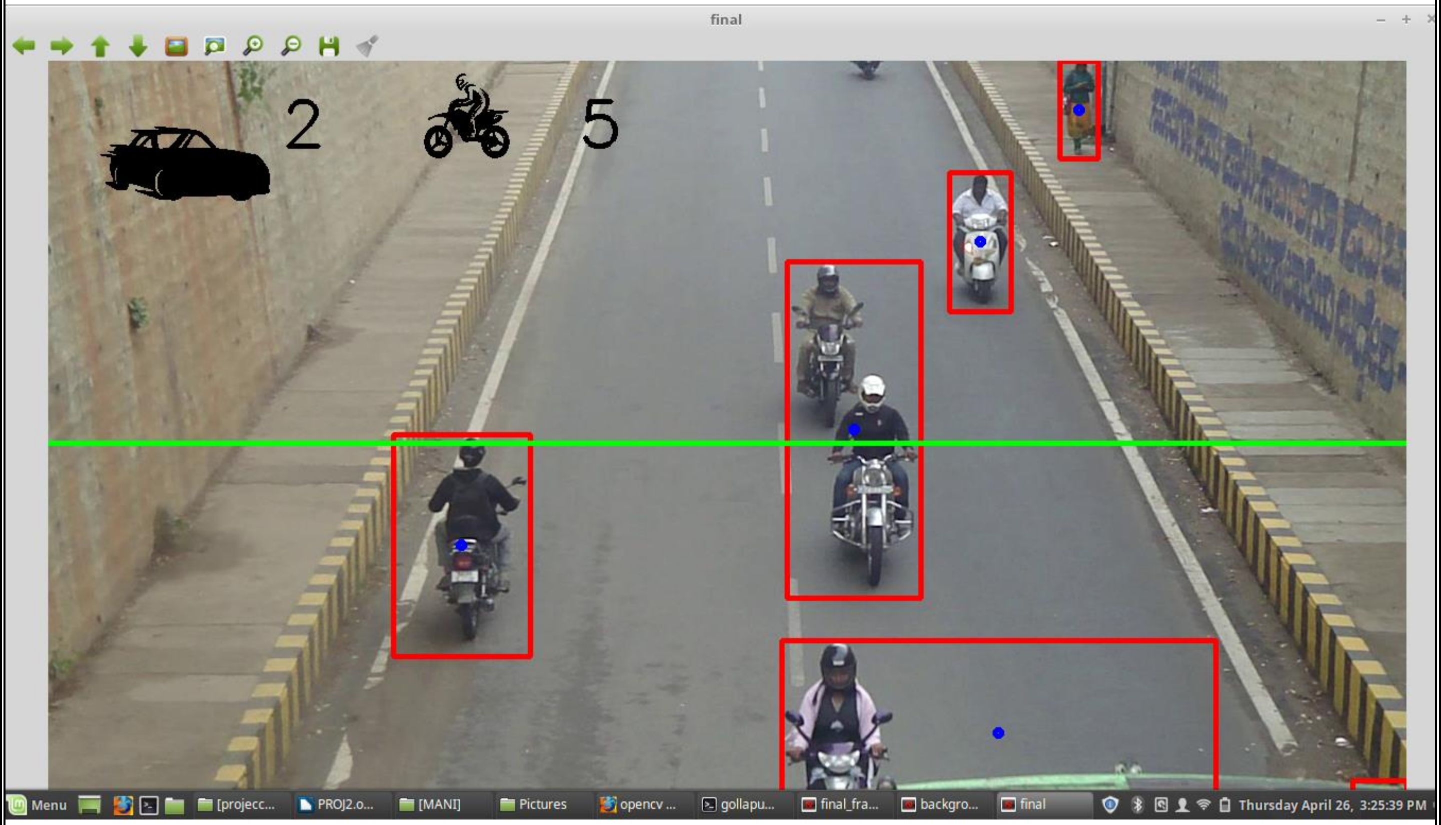
```
gollapudi@RKG-PC ~/Desktop $ cd ..
gollapudi@RKG-PC ~ $ cd Desktop
gollapudi@RKG-PC ~/Desktop $ python3 main.py
```

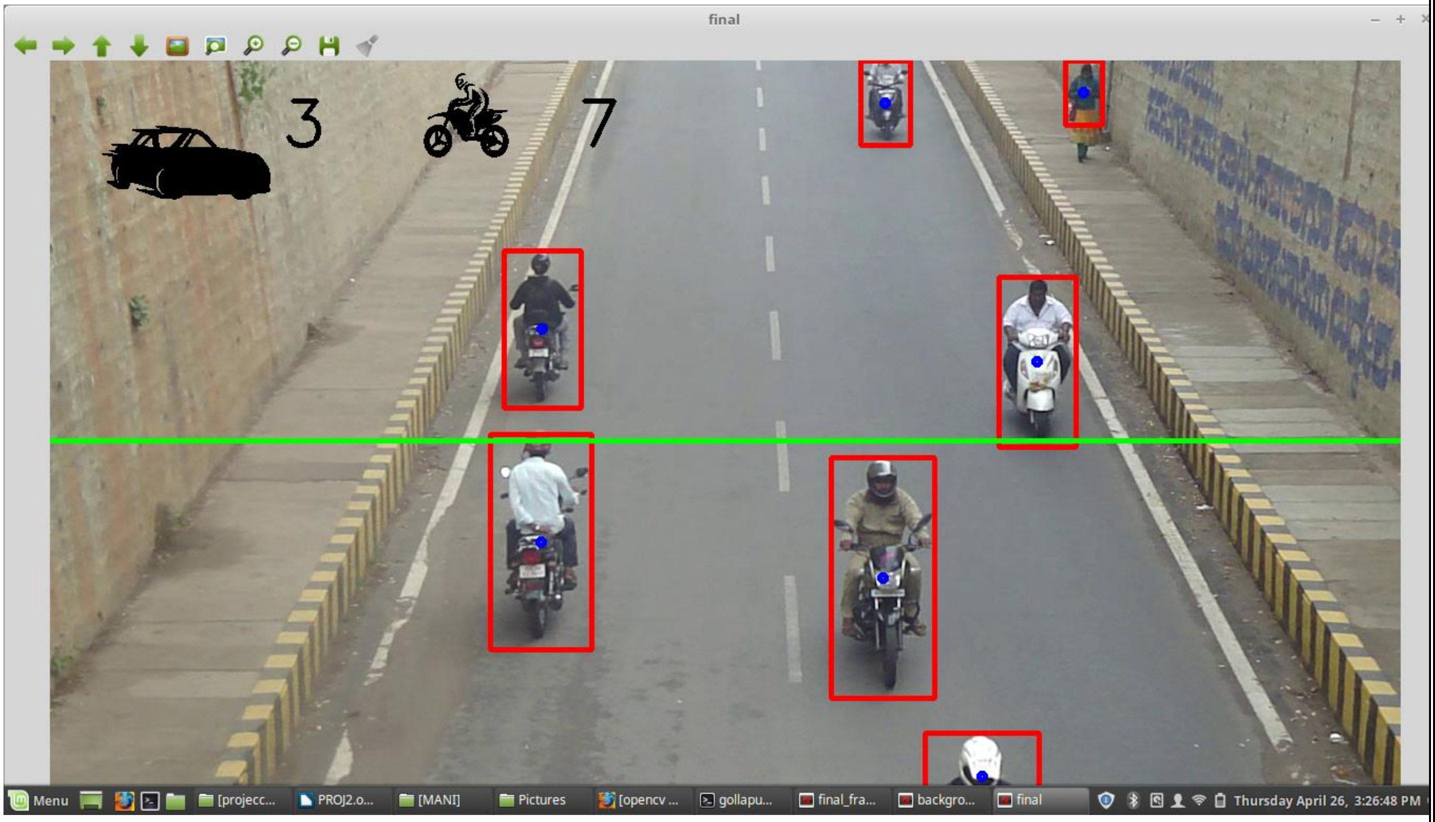
Menu
gollapudi@RKG-...
[projecctcode]
Mozilla Firefox
[projecctcode]
PROJ2.odt - Libre...
MANI
Thursday April 26, 2:50:35 PM



gollapudi@RKG-PC ~/Desktop
File Edit View Search Terminal Help

```
Initialising 93 %

Initialising 93 %

Initialising 93 %

Initialising 94 %

Initialising 94 %

Initialising 94 %

Initialising 95 %

Initialising 95 %

Initialising 95 %

Initialising 96 %

Initialising 96 %

Initialising 96 %

Initialising 97 %

Initialising 97 %

Initialising 97 %

Initialising 98 %

Initialising 98 %

Initialising 98 %

Initialising 99 %

Initialising 99 %

Initialising 99 %
Initialising 100 %
Segmentation fault (core dumped)
gollapudi@RKG-PC ~/Desktop $
```


final
2
5
Menu
[projecc...
PROJ2.o...
[MANI]
Pictures
opencv ...
gollapu...
final_fra...
backgro...
final
Thursday April 26, 3:25:39 PM


final
3
7
Menu
[projecc...
PROJ2.o...
[MANI]
Pictures
[opencv ...
gollapu...
final_fra...
backgro...
final
Thursday April 26, 3:26:48 PM

CONCLUSION:

Vehicle Detection and Counting is necessary to establish an enriched in formation platform and improve the quality of intelligent transportation systems. This solution for Vehicle Detection, Classification and Counting can be used in traffic monitoring, parking area allocation etc. A system has been developed to detect and count dynamic vehicles on highways efficiently. The experimental results show that the accuracy of counting vehicles was 90% with all the false positives involved. Complex algorithms can be employed in the future which focus on feature-based detection of vehicles to handle occlusion and light related issues. Researchers are also working for detection of vehicles using the headlights and the tail lights. By using this SMATRACON technology we can detect the vehicles count and can explore the traffic density easily.

FUTURE SCOPE:

Using the SMATRACON technology we can measure the traffic density easily. This will easily provide the scope of detecting the location of emergency ambulance in traffic areas. By tracking the location of ambulance, we can easily provide way for it. For identification of theft vehicles, it is the best platform such that the vehicle can be traced easily under any circumstances.

REFERENCES:

1) http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6112843

2) https://www.behance.net/gallery/Vehicle-Detection-Tracking-and-Counting/4057777

3) Huei-Yung Lin and Juang-Yu Wei, "A Street Scene Surveillance System for Moving Object Detection, Tracking and Classification", Proceedings of the IEEE Intelligent Vehicles Symposium Istanbul, Turkey, June 13-15, 2007.

4) F. Bardet, et al., "Unifying real-time multi-vehicle tracking and categorization," in Intelligent Vehicles Symposium, 2009 IEEE, 2009, pp. 197-202.

5) "OpenCV-Python Tutorials Documentation" Release 1,Alexander Mordvintsev & Abid K, Nov 05, 2017