# VIGNAN’S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH (Deemed to be UNIVERSITY)

**VADLAMUDI – 522 213, GUNTUR DIST, ANDHRA PRADESH, INDIA.**



**CERTIFICATE**

This is to certify that the Internship Report entitled **“DENSITY BASED SMART TRAFFIC CONTROL SYSTEM”** that is being submitted by **G.V.D.SAI.SIVA.RAO(171FD01025)** in partial fulfilment for the award of MCA post garduation in Information Technology at Vignan’s Foundation for Science, Technology and Research, Deemed to be University, is a record of bonafide work carried out by them at **“TECHNO ENDEAVOURS”** under the supervision of **“G.SRIDHAR”** under the co-guidance of the following faculty member of Department of IT.

**Project Guide** **Head of the department**

M.SriKanth Yadav Dr.K.V.KRISHNA KISHORE

Department of IT Department of IT

**DECLARATION**

I hereby declare that the project entitled “**DENSITY BASED SMART TRAFFIC CONTROL SYSTEM**” submitted to the **DEPARTMENT OF INFORMATION TECHNOLOGY**. This REPORT is our original work and the project has not formed the basis for the award of any degree, associate-ship, and fellowship or any other similar titles and no part of it has been published or sent for publication at the time of submission.

By

Date: 15-April 2020 G.V.D.Sai Siva Rao(171FD01025)

# ACKNOWLEDGMENT

Internship program is a golden opportunity for learning and self-development. Consider myself very lucky and honoured to have so many wonderful people lead me through in the completion of this project.

I express my gratitude towards the Managing Director of **G.Sridhar,** for permitting me to undertake the **Internship Program** in their extreme company and for the help and cooperation throughout the course of my Internship Program.

We feel it our responsibility to thank “**M.Srikanth Yadav**” under whose valuable guidance that the project came out successfully after each stage, and also it is our responsibility to extend our thanks to **Dr. P.Subba Rao, Department Internship Coordinator,** for extending his support towards the Internship Program in **“Techno Endeavours Pvt.Ltd”**, in **Hyderabad**.

It is a great pleasure for me to express my sincere thanks to **Prof. K. V. Krishna Kishore, HOD, IT** of VFSTR Deemed to be University**,** for providing me an opportunity to do my Internship Program.

We extend our wholehearted gratitude to all our faculty members of Department of Information Technology who helped us in our academics throughout course.

Finally, we wish to express thanks to our family members for the love and affection overseas and forbearance and cheerful depositions, which are vital for sustaining effort, required for completing this work.

With Sincere regards,

G.V.D.Sai Siva Rao (171FD01025)

|  |  |  |  |
| --- | --- | --- | --- |
| **TABLE OF CONTENTS** | | | |
|  |  | ***Title*** | ***Page No.*** |
|  | **TABLE OF CONTENTS** | |  |
| **1** | **INTRODUCTION** | |  |
|  | 1.1 Project Idea | | 11 |
| **2** | **SOFTWARE REQUIREMENT SPECIFICATION** | | |
|  | 2.1 Introduction | | 12 |
|  | 2.2 Functional Requirements | | 13 |
|  | 2.3 Non Functional Requirements | | 13 |
|  | 2.4 | Hardware Requirements | 13 |
|  | 2. 5 Software Requirements | | 13 |
| **3** | **ANALYSIS AND DESIGN PHASE** | |  |
|  | 3.1 Introduction | | 14 |
|  | 3.2 Use Case Diagram | | 14 |
|  | 3.3 | Class Diagram | 15 |
|  | 3.4 | Sequence Diagram | 15 |
|  | 3.5 | Activity Diagram | 16 |
| 3.6 State Diagram | | | 17 |
|  | 3.7 Deployment Diagram | | 18 |
|  | 3.8 Data Design | | 18 |
|  | 3.9 Conclusion | | 19 |

|  |  |
| --- | --- |
| **4 SYSTEM LOW LEVEL DESIGN** |  |
| 4.1 Modules of the project | 20-21 |
| **5 IMPLEMENTATION** |  |
| 5.1 Same Code | 22-28 |
| 5.2 Screen Captures | 29-32 |
|  |  |
| **6 TESTING** |  |
| 6.1 Software Testing | 32 |
| 6.2 Black Box Testing | 32 |
| 6.3 White Box Testing | 32 |
| 6.4 Performance Testing | 32 |
| 6.5 Load Testing | 33 |
| 6.6 Manual Testing | 34 |
|  |  |
| **7 RESULTS AND CHALLENGES** |  |
| 7.1 Results | 36 |
| 7.2 Challenges | 36 |
| **8 CONCLUSION AND FUTURE WORK** |  |
| 8.1 Conclusion | 37 |
| 8.2 Scope for future work | 37 |
|  |  |
| **9 REFERENCE** | 38 |

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **TITLE** | **PAGENO** |
| 3.1 | Use Case Diagram | 15 |
| 3.2 | Class Diagram | 16 |
| 3.3 | Activity Diagram | 17 |
| 3.4 | Sequence diagram | 16 |
| 3.5 | State Chart diagram | 18 |
| 3.6 | Deployment diagram | 19 |
| 5.2.1 | User upload screen | 30 |
| 5.2.2 | Upload image page | 31 |
| 5.2.3 | Reference image | 32 |
| 5.2.4 | Both images | 34 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| Table 3.10.1 | SQLite Database |  |
| Table 3.10.2 | List of tables in database |  |
| Table 6:1 | Test case for empty login fields |  |
| Table 6:2 | Test case for wrong login fields |  |
| Table 6:3 | Test case for signup fail |  |

**ACRONYMS & ABBREVIATIONS**

|  |  |
| --- | --- |
| * **HTML:** | Hyper Text Markup Language. |
| * **XML:** | Extensible Markup Language. |
| * **IDE:** | Integrated Development Environment |
| * **GUI:** | Graphical User Interface |
| * **HTTP:** | Hyper Text Transfer Protocol |
| * **API:** | Application Programming Interface |
| * **E-R:** | Entity-Relationship |
| * **UML:** | Unified Modeling Language |
| * **OOAD:** | Object-Oriented Analysis & Design. |

**INTERNSHIP SUMMARY**

**Location:**Hyderabad

**Center:** “Techno Endeavours”

**Duration:** 6 months

**Date of start:**11-12-2019

**Date of submission:**15-4-2020

**Title of project:**Density Based Smart Traffic Control System

**Team Members:**2

G.V.D.Sai Siva Rao(171FD01025)

Ch Siva Nagarjuna Reddy(171FD01029)

**Name of the guide**:G.Sridhar

**Name of Faculty guide:**M.SriKanth Yadav VFSTR University. **Project Area:** Traffic Control System

**Abstract:**

In this paper author is describing concept to control or automate green traffic signal allotment time based on congestion available at road side using Canny Edge Detection Algorithm. To implement this technique we are uploading current traffic image to the application and application will extract edges from images and if there is more traffic then there will be more number of edges with white colour and if uploaded image contains less traffic then it will have less number of white colour edges. Empty edges will have black colour with value 0. By counting number of non-zeroes white pixels we will have complete idea of available traffic and based on that we will allocate time to green signal. If less traffic is there then green signal time will be less otherwise green signal allocation time will be more.

**Signature of Student Signature of Faculty Guide**

**Date: Date:**

**PROFILE OF THE COMPANY**

**About Techno Endeavours**

**Techno Endeavours :**Techno Endeavours was established in 2004. Techno Endeavours is a well reputed as Application solution providers in Hyderabad, India. The software development division was setup to support our US and offshore clients. We design, develop and deliver custom solutions that are usable, innovative and cost effective. Our experienced team excels in executing projects of any intensity for our clients across the world. Techno Endeavours was established with a vision to be a recognized leader in providing reliable business solution through computing and creative excellence. Committed to delivering the best solution for each client, we fuse technology with business strategy to address our clients’ current needs and position them for future opportunities. Our expertise and experience, teamed with our extensive knowledge in business application development, allows Techno Endeavours to deliver the synergy of creativity and user friendly systems on all platforms to our clients.

**Company address:**

17/B,adagutta,beside Arjun theater

KPHB, Hyderabad

**CHAPTER 1**

**INTRODUCTION**

Traffic congestion is one of the major modern-day crisis in every big city in the world.Intermetropolitan area studies suggest that traffic congestion reduces regional competitiveness and redistributes economic activity by slowing growth in county gross output or slowing metropolitan area employment growth .As more and more vehicles are commissioning in an already congested traffic system, there is an urgent need for a whole new traffic control system using advanced technologies to utilize the already existent infrastructures to its full extent.

Since building new roads, flyovers, elevated expressway etc. needs extensive planning, huge capital and lots of time; focus should be directed upon availing existing infrastructures more efficiently and diligently. glean traffic data. Some of them count total number of pixels [3], some of the work calculate number of vehicles . -rickshaw as vehicles which are the quotidian means of traffic especially in South-Asian countries. But this may be disadvantageous for those who are in lanes that have less frequency of traffic.

## 

**CHAPTER 2**

**REQUIREMENT ANALYSIS**

**Project Overview :-**

In this paper author is describing concept to control or automate green traffic signal allotment time based on congestion available at road side using Canny Edge Detection Algorithm. To implement this technique we are uploading current traffic image to the application and application will extract edges from images and if there is more traffic then there will be more number of edges with white colour and if uploaded image contains less traffic then it will have less number of white colour edges. Empty edges will have black colour with value 0. By counting number of non-zeroes white pixels we will have complete idea of available traffic and based on that we will allocate time to green signal. If less traffic is there then green signal time will be less otherwise green signal allocation time will be more. To compare current traffic we will take one reference image with high traffic and comparison will be done between uploaded image white pixels and reference image white pixels. Using below code we will allocate time to green signal.

**Existing System :-**

Edge detection technique is imperative to extract the required traffic information from the CCTV footage. It can be used to isolate the required information from rest of the image. There are several edge detection techniques available. They have distinct characteristics in terms of noise reduction, detection sensitivity, accuracy etc. Among them, Prewitt , canny ,Sobel, Roberts and LOG are most accredited operators. It has been observed that the Canny edge detector depicts higher accuracy in detection of object with higher entropy, PSNR(Peak Signal to Noise Ratio), MSE(Mean Square Error) and execution time compared with Sobel, Roberts, Prewitt, Zero crossing and LOG .Here is a comparison between distinct edge detection techniques.

To implement this technique we are uploading current traffic image to the application and application will extract edges from images and if there is more traffic then there will be more number of edges with white colour and if uploaded image contains less traffic then it will have less number of white colour edges.

**Proposed System :-**

In this paper, a system in which density of traffic is measured by comparing captured image with real time traffic information against the image of the empty road as reference image is proposed. Here, in figure 1, the block diagram for proposed traffic control technique is illustrated.

Each lane will have a minimum amount of green signal duration allocated. According to the percentage of matching allocated traffic light duration can be controlled. The matching is achieved by comparing the number of white points between two images. The entire image processing before edge detection i.e. image acquisition, image resizing, RGB to gray conversion and noise reduction is explained in section II. At section III, canny edge detection operation and white point count are depicted. Canny edge detector operator is selected because of its greater overall performance.

**Advantages :-**

It is advantageous to convert RGB images into grayscale for further processing. When converting an RGB image to grayscale, it is pertinent to consider the RGB values for each pixel and make as output a single value reflecting the brightness of that pixel. One of the approaches isto take the average of the contribution from each channel:(R+B+C)/3.

**Functional requirements :-**

In software engineering, a functional requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

**Non –Functional Requirements :-**

A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?

A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

**Hardware Requirements :-**

* Operating System supported by

1. Windows 7

2. Windows XP

3 . Windows 8

* Processor – Pentium IV or higher
* RAM -- 256 MB
* Space on Hard Disk -- Minimum 512 MB

**Software Requirements :-**

* For developing the Application

1. Python

2. Django

3. Mysql

4. Mysqlclient

5. WampServer 2.4

* Technologies and Languages used to Develop

-- Python

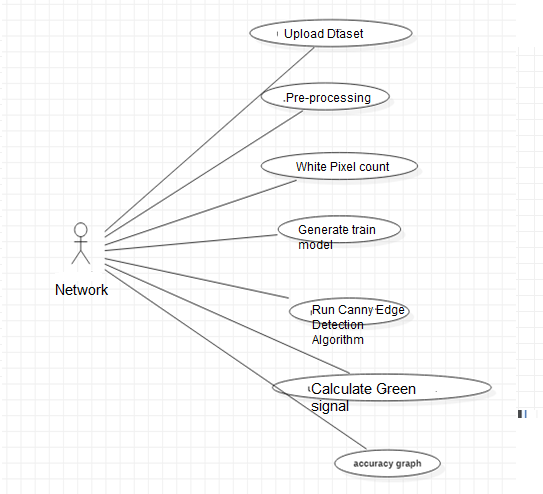
**CHAPTER 3**

**DESIGN PHASE**

**INTRODUCTION**

This chapter provides the design phase of the Application. To design the project, we use the UML diagrams. The Unified Modelling Language (UML) is a general- purpose, developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

### 3.1 USE CASE DIAGRAM

****

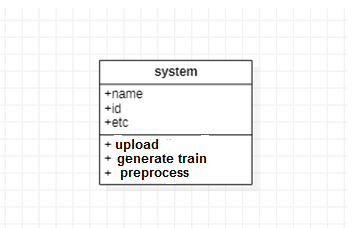
**Fig 2.1 Use case Diagram**

The use case diagram is used to represent all the functional use cases that are involved in the project.

The above diagram represents the main two **actors** in the project, they are

* + - User

### CLASS DIAGRAM

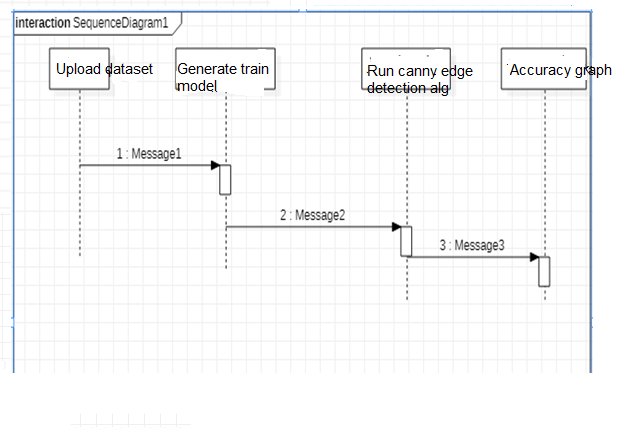


**Fig 3.2 class diagram**

The above mentioned class diagram represents the Chatbot system workflow model. This diagram has class models with class names as

* + - User
    - Home screen

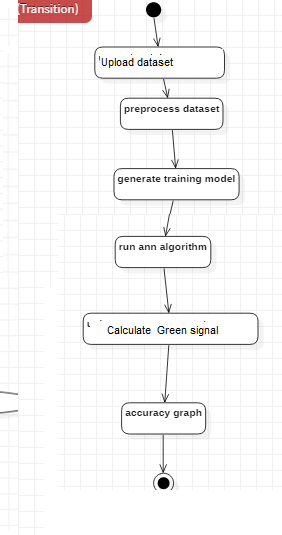
### SEQUENCE DIAGRAM

****

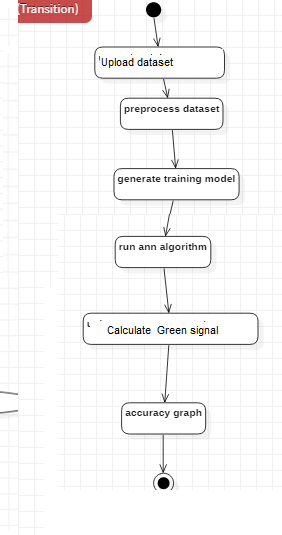
**Fig 3.5 sequence diagram**

The above diagram represents the sequence of flow of actions in the system.

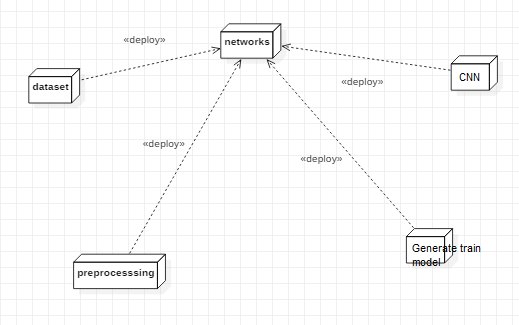
### ACTIVITY DIAGRAM

****

**3.5 STATE CHART DIAGRAM :-**

****

**3.6 DEPLOYEMENT DIAGRAM :-**

****

## DATA DESIGN

* + 1. **Databases SQLite**

|  |
| --- |
| **Name** |
| Density based smart traffic |

**Table 3.10.1 SQLite Database**

* + 1. **Tables**

|  |  |
| --- | --- |
| **Name** | **Description** |
| Users | Contains all the registered user details. |
| View upload data sets | All the registered service provider details. |
| Services | Contains all the types of services available. |

**Table 3.10.2 List of Database Tables**

## 3.5 CONCLUSION

In this paper, a smart traffic control system availing image processing as an instrument for measuring the density has been proposed. Besides explaining the limitations of current near obsolete traffic control system, the advantages of proposed traffic control system have been demonstrated. For this purpose, four sample images of different traffic scenario have been attained. Upon completion of edge detection, the similarity between sample images with the reference image has been calculated. Using this similarity, time allocation has been carried out for each individual image in accordance with the time allocation algorithm. In addition, similarity in percentage and time allocation have been illustrated for each of the four sample images using Python programming language. Besides presenting the schematics for the proposed smart traffic control system, all the necessary results have been verified by hardware implementation.

### CHAPTER 4

**SYSTEM LOWLEVEL DESIGN**

**4.1 Modules of the Application:**

**Upload Image Module:**

In this module current traffic image will be uploaded to application and then convert colourimage into Gray Scale image format to have pixels values as black and white colour.

**Pre-process module:**

In this module Gaussian Filter will be applied on uploaded image to convert image into smooth format. After applying filter Canny Edge Detection will be applied on image to get edges from the image. Each vehicle will have white colour pixels and non-vehicle will have black colour pixels.

**White Pixel Count Module:**

Using this module we will count white pixels from canny image to get complete

traffic count

**Calculate Green Signal Time Allocation Module:**

Based on white pixel count traffic signal time will be calculated. How this time

will be calculated is already explain in previous page.

**CANNY EDGE DETECTION ALGORITHM:**

A lot of people consider the Canny Edge Detector the ultimate edge detector. You get

clean, thin edges that are well connected to nearby edges. If you use some image processing

package,you probably get a function that does everything. Here, I'll go into exactly how they

work.

The canny edge detector is a multistage edge detection algorithm. The steps are:

**Pre-processing**

**Calculating gradients**

**Non-maximum suppression**

**Thresholding with hysterysis**

The two key parameters of the algorithm are - an upper threshold and a lower threshold. The

upper threshold is used to mark edges that are definitely edges. The lower threshold is to find

faint pixels that are actually a part of an edge.

The general criteria for edge detection include:

Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible.The edge point detected from the operator should accurately localize on the center of the edge.

A given edge in the image should only be marked once, and where possible, image noise

should not create false edges.

To satisfy these requirements Canny used the calculus of variations – a technique which finds the function which optimizes a given functional. The optimal function in Canny's detector is described by the sum of four exponential terms, but it can be approximated by the first derivative of a Gaussian.Among the edge detection methods developed so far, Canny edge detection algorithm is one of the most strictly defined methods that provides good and reliable detection. Owing to its optimality to meet with the three criteria for edge detection and the simplicity of process for implementation, it became one of the most popular algorithms for edge detection.

**CHAPTER 5**

**IMPLEMENTATION**

* 1. **Sample Code**
     1. **Python Code**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

import numpy as np

from tkinter.filedialog import askopenfilename

import numpy as np

from CannyEdgeDetector import \*

import skimage

import matplotlib.image as mpimg

import os

import scipy.misc as sm

import cv2

import matplotlib.pyplot as plt

main = tkinter.Tk()

main.title("Density Based Smart Traffic Control System")

main.geometry("1300x1200")

global filename

global refrence\_pixels

global sample\_pixels

def rgb2gray(rgb):

r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]

gray = 0.2989 \* r + 0.5870 \* g + 0.1140 \* b

return gray

def uploadTrafficImage():

global filename

filename = filedialog.askopenfilename(initialdir="images")

pathlabel.config(text=filename)

def visualize(imgs, format=None, gray=False):

j = 0

plt.figure(figsize=(20, 40))

for i, img in enumerate(imgs):

if img.shape[0] == 3:

img = img.transpose(1,2,0)

plt\_idx = i+1

plt.subplot(2, 2, plt\_idx)

if j == 0:

plt.title('Sample Image')

plt.imshow(img, format)

j = j + 1

elif j > 0:

plt.title('Reference Image')

plt.imshow(img, format)

plt.show()

def applyCanny():

imgs = []

img = mpimg.imread(filename)

img = rgb2gray(img)

imgs.append(img)

edge = CannyEdgeDetector(imgs, sigma=1.4, kernel\_size=5, lowthreshold=0.09, highthreshold=0.20, weak\_pixel=100)

imgs = edge.detect()

for i, img in enumerate(imgs):

if img.shape[0] == 3:

img = img.transpose(1,2,0)

cv2.imwrite("gray/test.png",img)

temp = []

img1 = mpimg.imread('gray/test.png')

img2 = mpimg.imread('gray/refrence.png')

temp.append(img1)

temp.append(img2)

visualize(temp)

def pixelcount():

global refrence\_pixels

global sample\_pixels

img = cv2.imread('gray/test.png', cv2.IMREAD\_GRAYSCALE)

sample\_pixels = np.sum(img == 255)

img = cv2.imread('gray/refrence.png', cv2.IMREAD\_GRAYSCALE)

refrence\_pixels = np.sum(img == 255)

messagebox.showinfo("Pixel Counts", "Total Refrence White Pixels Count : "+str(sample\_pixels)+"\nTotal Sample White Pixels Count : "+str(refrence\_pixels))

def timeAllocation():

avg = (sample\_pixels/refrence\_pixels) \*100

if avg >= 90:

messagebox.showinfo("Green Signal Allocation Time","Traffic is very high allocation green signal time : 60 secs")

if avg > 85 and avg < 90:

messagebox.showinfo("Green Signal Allocation Time","Traffic is high allocation green signal time : 50 secs")

if avg > 75 and avg <= 85:

messagebox.showinfo("Green Signal Allocation Time","Traffic is moderate green signal time : 40 secs")

if avg > 50 and avg <= 75:

messagebox.showinfo("Green Signal Allocation Time","Traffic is low allocation green signal time : 30 secs")

if avg <= 50:

messagebox.showinfo("Green Signal Allocation Time","Traffic is very low allocation green signal time : 20 secs")

def exit():

main.destroy()

font = ('times', 16, 'bold')

title = Label(main, text=' Density Based Smart Traffic Control System Using Canny Edge Detection Algorithm for Congregating Traffic Information',anchor=W, justify=CENTER)

title.config(bg='yellow4', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 14, 'bold')

upload = Button(main, text="Upload Traffic Image", command=uploadTrafficImage)

upload.place(x=50,y=100)

upload.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='yellow4', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=50,y=150)

process = Button(main, text="image preprocessing using canny edge detection" ,command=applyCanny)

process.place(x=50,y=200)

process.config(font=font1)

count = Button(main, text="white pixel count",command=pixelcount)

count.place(x=50,y=250)

count.config(font=font1)

count = Button(main, text="calculate green signal time allocation",command=timeAllocation)

count.place(x=50,y=300)

count.config(font=font1)

exitButton = Button(main, text="exit",command=exit)

exitButton.place(x=50,y=350)

exitButton.config(font=font1)

main.config(bg='magenta3')

main.mainloop()

**5.1.2 Canny Edge Detection Alogithm**

from scipy import ndimage

from scipy.ndimage.filters import convolve

from scipy import misc

import numpy as np

class CannyEdgeDetector:

def \_\_init\_\_(self, imgs, sigma=1, kernel\_size=5, weak\_pixel=75, strong\_pixel=255, lowthreshold=0.05, highthreshold=0.15):

print(imgs)

self.imgs = imgs

self.imgs\_final = []

self.img\_smoothed = None

self.gradientMat = None

self.thetaMat = None

self.nonMaxImg = None

self.thresholdImg = None

self.weak\_pixel = weak\_pixel

self.strong\_pixel = strong\_pixel

self.sigma = sigma

self.kernel\_size = kernel\_size

self.lowThreshold = lowthreshold

self.highThreshold = highthreshold

return

def gaussian\_kernel(self, size, sigma=1):

size = int(size) // 2

x, y = np.mgrid[-size:size+1, -size:size+1]

normal = 1 / (2.0 \* np.pi \* sigma\*\*2)

g = np.exp(-((x\*\*2 + y\*\*2) / (2.0\*sigma\*\*2))) \* normal

return g

def sobel\_filters(self, img):

Kx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], np.float32)

Ky = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]], np.float32)

Ix = ndimage.filters.convolve(img, Kx)

Iy = ndimage.filters.convolve(img, Ky)

G = np.hypot(Ix, Iy)

G = G / G.max() \* 255

theta = np.arctan2(Iy, Ix)

return (G, theta)

def non\_max\_suppression(self, img, D):

M, N = img.shape

Z = np.zeros((M,N), dtype=np.int32)

angle = D \* 180. / np.pi

angle[angle < 0] += 180

for i in range(1,M-1):

for j in range(1,N-1):

try:

q = 255

r = 255

if (0 <= angle[i,j] < 22.5) or (157.5 <= angle[i,j] <= 180):

q = img[i, j+1]

r = img[i, j-1]

elif (22.5 <= angle[i,j] < 67.5):

q = img[i+1, j-1]

r = img[i-1, j+1]

elif (67.5 <= angle[i,j] < 112.5):

q = img[i+1, j]

r = img[i-1, j]

elif (112.5 <= angle[i,j] < 157.5):

q = img[i-1, j-1]

r = img[i+1, j+1]

if (img[i,j] >= q) and (img[i,j] >= r):

Z[i,j] = img[i,j]

else:

Z[i,j] = 0

except IndexError as e:

pass

return Z

def threshold(self, img):

highThreshold = img.max() \* self.highThreshold;

lowThreshold = highThreshold \* self.lowThreshold;

M, N = img.shape

res = np.zeros((M,N), dtype=np.int32)

weak = np.int32(self.weak\_pixel)

strong = np.int32(self.strong\_pixel)

strong\_i, strong\_j = np.where(img >= highThreshold)

zeros\_i, zeros\_j = np.where(img < lowThreshold)

weak\_i, weak\_j = np.where((img <= highThreshold) & (img >= lowThreshold))

res[strong\_i, strong\_j] = strong

res[weak\_i, weak\_j] = weak

return (res)

def hysteresis(self, img):

M, N = img.shape

weak = self.weak\_pixel

strong = self.strong\_pixel

for i in range(1, M-1):

for j in range(1, N-1):

if (img[i,j] == weak):

try:

if ((img[i+1, j-1] == strong) or (img[i+1, j] == strong) or (img[i+1, j+1] == strong)

or (img[i, j-1] == strong) or (img[i, j+1] == strong)

or (img[i-1, j-1] == strong) or (img[i-1, j] == strong) or (img[i-1, j+1] == strong)):

img[i, j] = strong

else:

img[i, j] = 0

except IndexError as e:

pass

return img

def detect(self):

imgs\_final = []

for i, img in enumerate(self.imgs):

self.img\_smoothed = convolve(img, self.gaussian\_kernel(self.kernel\_size, self.sigma))

self.gradientMat, self.thetaMat = self.sobel\_filters(self.img\_smoothed)

self.nonMaxImg = self.non\_max\_suppression(self.gradientMat, self.thetaMat)

self.thresholdImg = self.threshold(self.nonMaxImg)

img\_final = self.hysteresis(self.thresholdImg)

self.imgs\_final.append(img\_final)

return self.imgs\_final

* + 1. **Test**

import cv2

import os

import numpy as np

from test1 import \*

import numpy as np

import skimage

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import os

import scipy.misc as sm

def rgb2gray(rgb):

r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]

gray = 0.2989 \* r + 0.5870 \* g + 0.1140 \* b

return gray

def visualize(imgs, format=None, gray=False):

plt.figure(figsize=(20, 40))

for i, img in enumerate(imgs):

if img.shape[0] == 3:

img = img.transpose(1,2,0)

plt\_idx = i+1

plt.subplot(2, 2, plt\_idx)

plt.imshow(img, format)

plt.show()

def auto\_canny(image, sigma=0.33):

# compute the median of the single channel pixel intensities

v = np.median(image)

# apply automatic Canny edge detection using the computed median

lower = int(max(0, (1.0 - sigma) \* v))

upper = int(min(255, (1.0 + sigma) \* v))

edged = cv2.Canny(image, lower, upper)

return edged

filename = 'images'

imgs = []

#for root, dirs, files in os.walk(filename):

# for fdata in files:

# img = mpimg.imread(root+'/'+fdata)

# img = rgb2gray(img)

# imgs.append(img)

#t = test1(root+'/'+fdata)

#img = t.detect()

#image = cv2.imread(root+'/'+fdata)

#image = cv2.resize(image, (400,400))

#image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

#image = cv2.GaussianBlur(image, (5, 5), 0)

#grad\_x = cv2.Sobel(gray, cv2.CV\_16S, 1, 0, ksize=3, scale=1, delta=0, borderType=cv2.BORDER\_DEFAULT)

#grad\_y = cv2.Sobel(gray, cv2.CV\_16S, 0, 1, ksize=3, scale=1, delta=0, borderType=cv2.BORDER\_DEFAULT)

#abs\_grad\_x = cv2.convertScaleAbs(grad\_x)

#abs\_grad\_y = cv2.convertScaleAbs(grad\_y)

#grad = cv2.addWeighted(abs\_grad\_x, 0.5, abs\_grad\_y, 0.5, 0)

#gray = cv2.Canny(image,25,255,L2gradient=False)#cv2.Canny(grad,100,200,L2gradient=True)

#cv2.imwrite("gray/"+fdata,img)

img = mpimg.imread('images/D.png')

img = rgb2gray(img)

imgs.append(img)

t = test1(imgs, sigma=1.4, kernel\_size=5, lowthreshold=0.09, highthreshold=0.20, weak\_pixel=100)

imgs = t.detect()

for i, img in enumerate(imgs):

if img.shape[0] == 3:

img = img.transpose(1,2,0)

cv2.imwrite("gray/D.png",img)

img = mpimg.imread('images/refrence.png')

img = rgb2gray(img)

imgs.append(img)

t = test1(imgs, sigma=1.4, kernel\_size=5, lowthreshold=0.09, highthreshold=0.20, weak\_pixel=100)

imgs = t.detect()

for i, img in enumerate(imgs):

if img.shape[0] == 3:

img = img.transpose(1,2,0)

cv2.imwrite("gray/refrence.png",img)

img = cv2.imread('gray/D.png', cv2.IMREAD\_GRAYSCALE)

pixel1 = np.sum(img == 255)

print('Number of white pixels:', pixel1)

img = cv2.imread('gray/refrence.png', cv2.IMREAD\_GRAYSCALE)

pixel2 = np.sum(img == 255)

print('Number of white pixels:', pixel2)

avg = (pixel1/pixel2) \*100

print(avg)

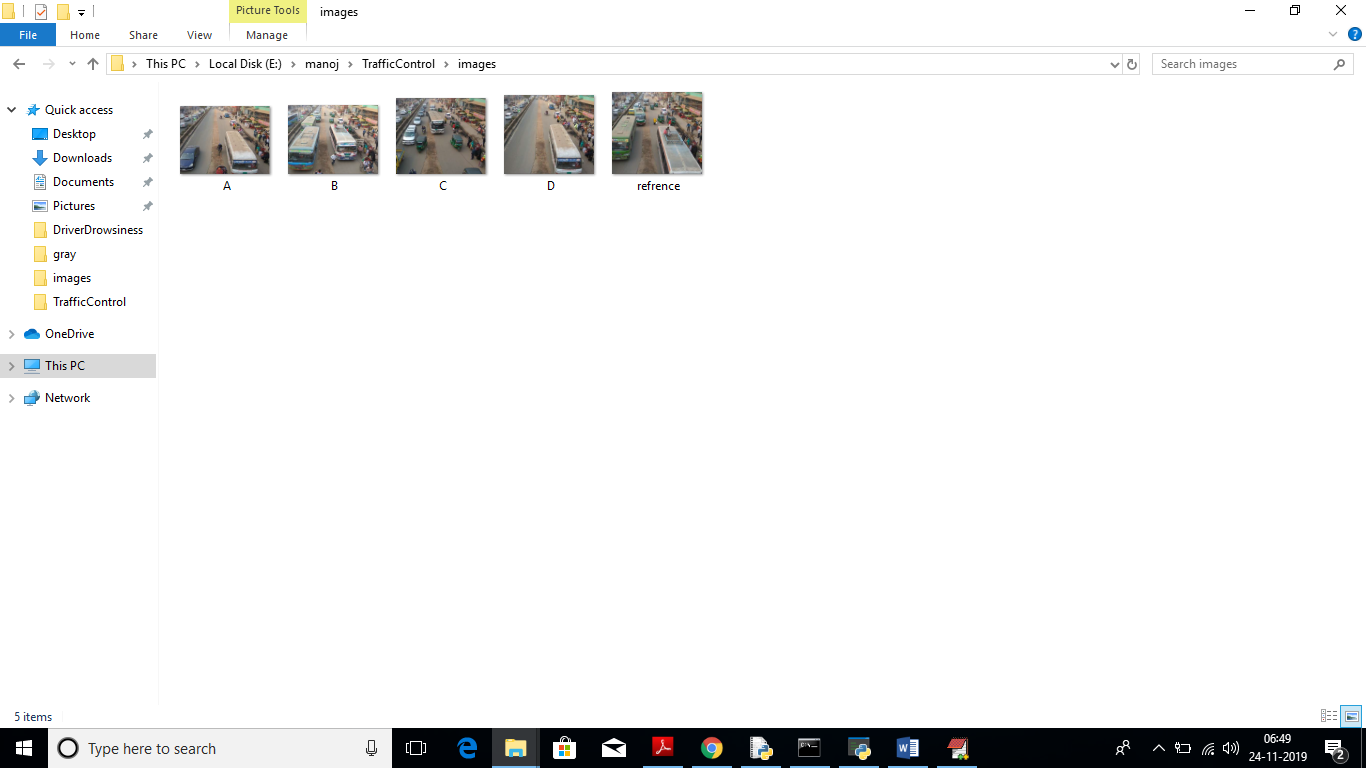
#cv2.waitKey(0)

#cv2.destroyAllWindows()

## Screen Captures

* + 1. **User Login Screen:**

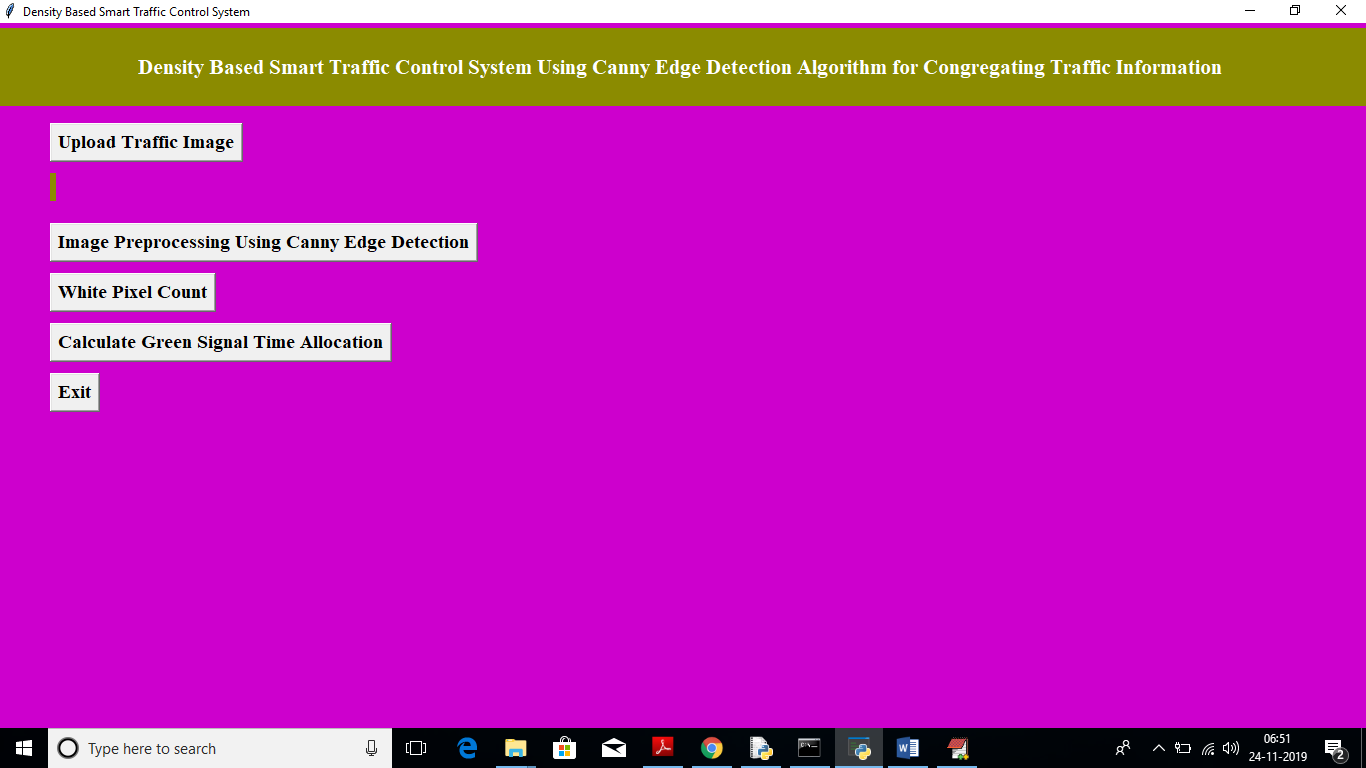
To implement this project we are using 4 input images given in paper and on reference image. Below are the images screen shots saved inside images folder



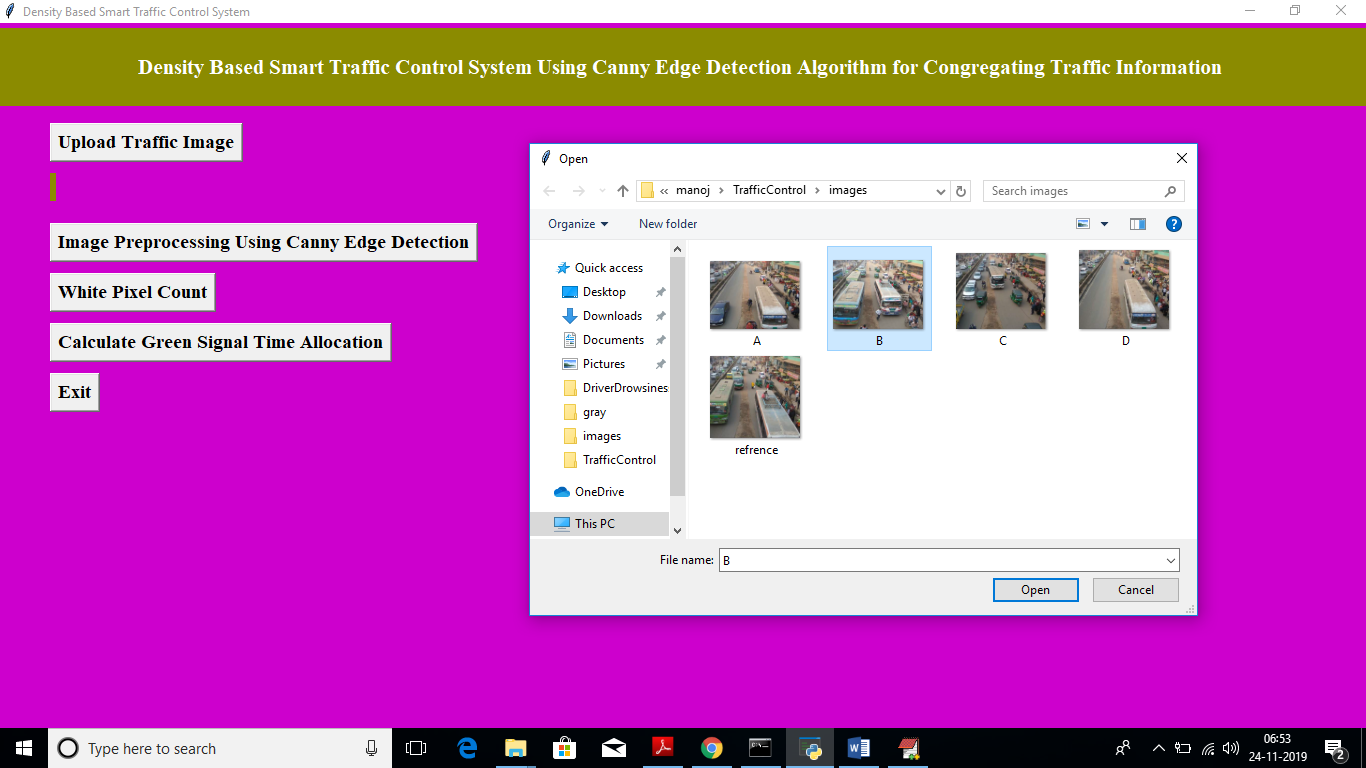
We can upload above 4 images to application to calculate traffic signal time.

Screen shots

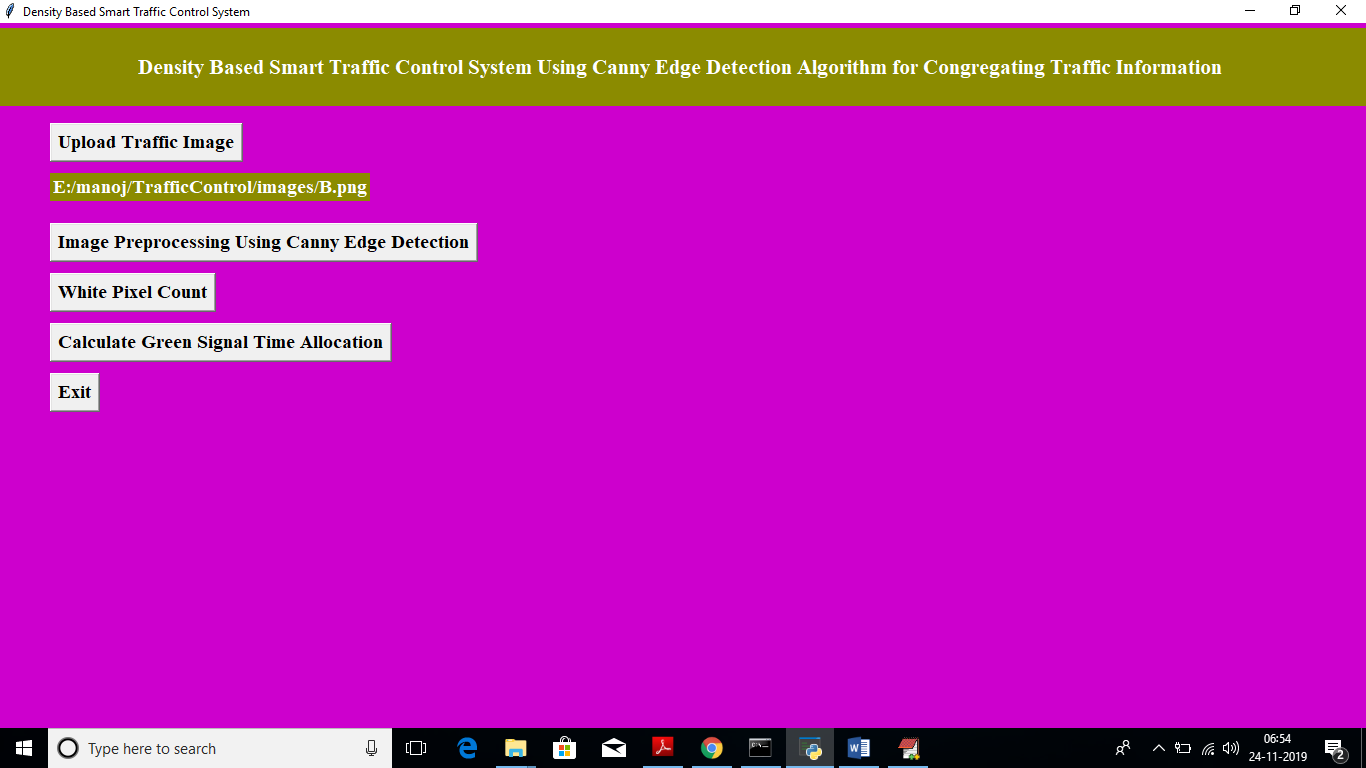
To run this project double click on ‘run.bat’ file to get below screen



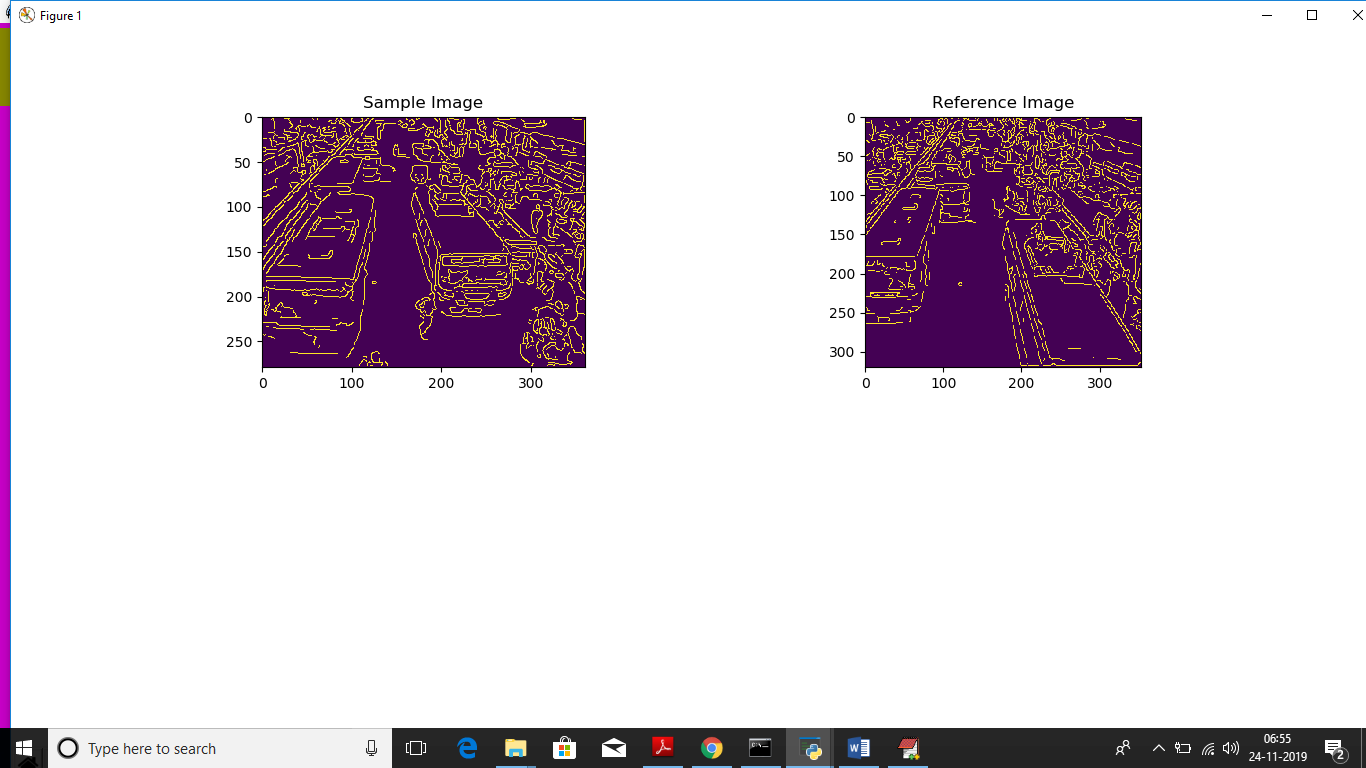
In above screen click on ‘Upload Traffic Image’ button to upload image.



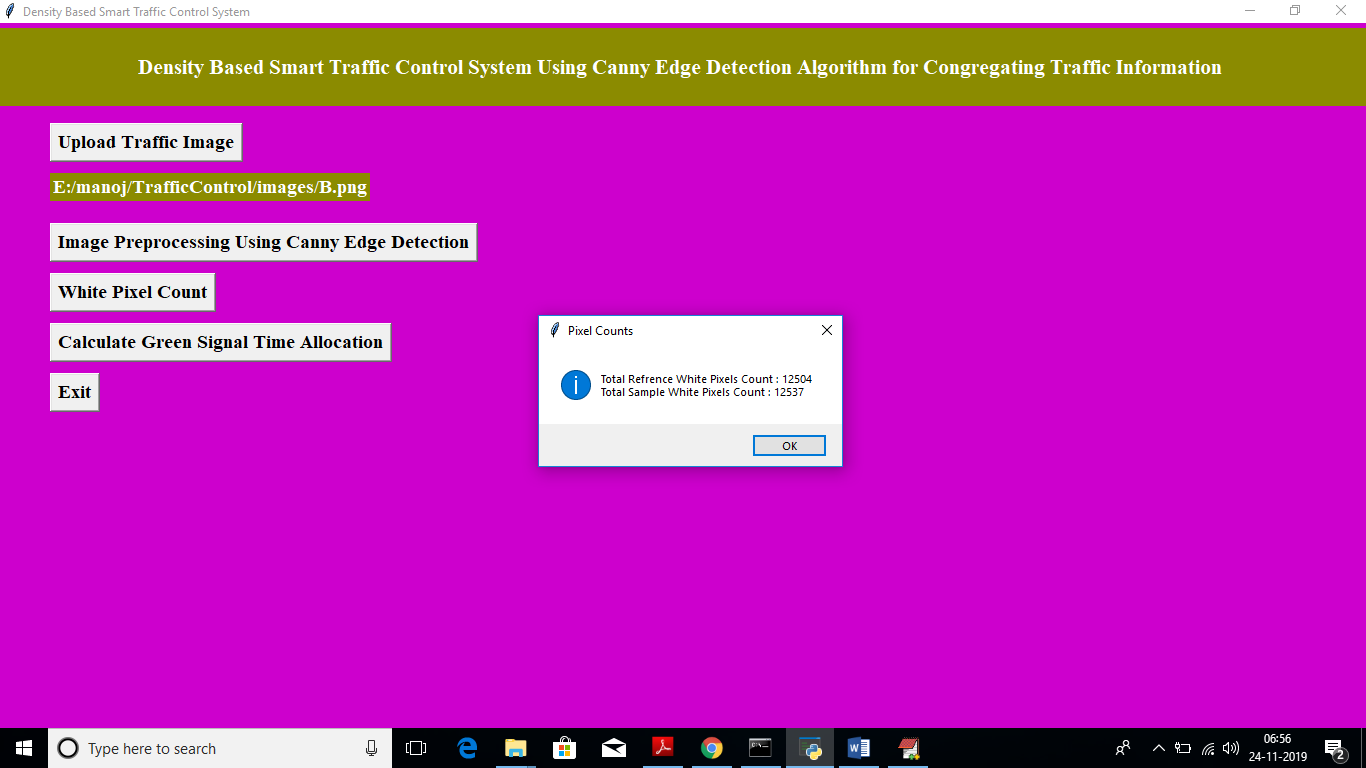
In above screen I am uploading image B and now click on ‘Open’ button to load image



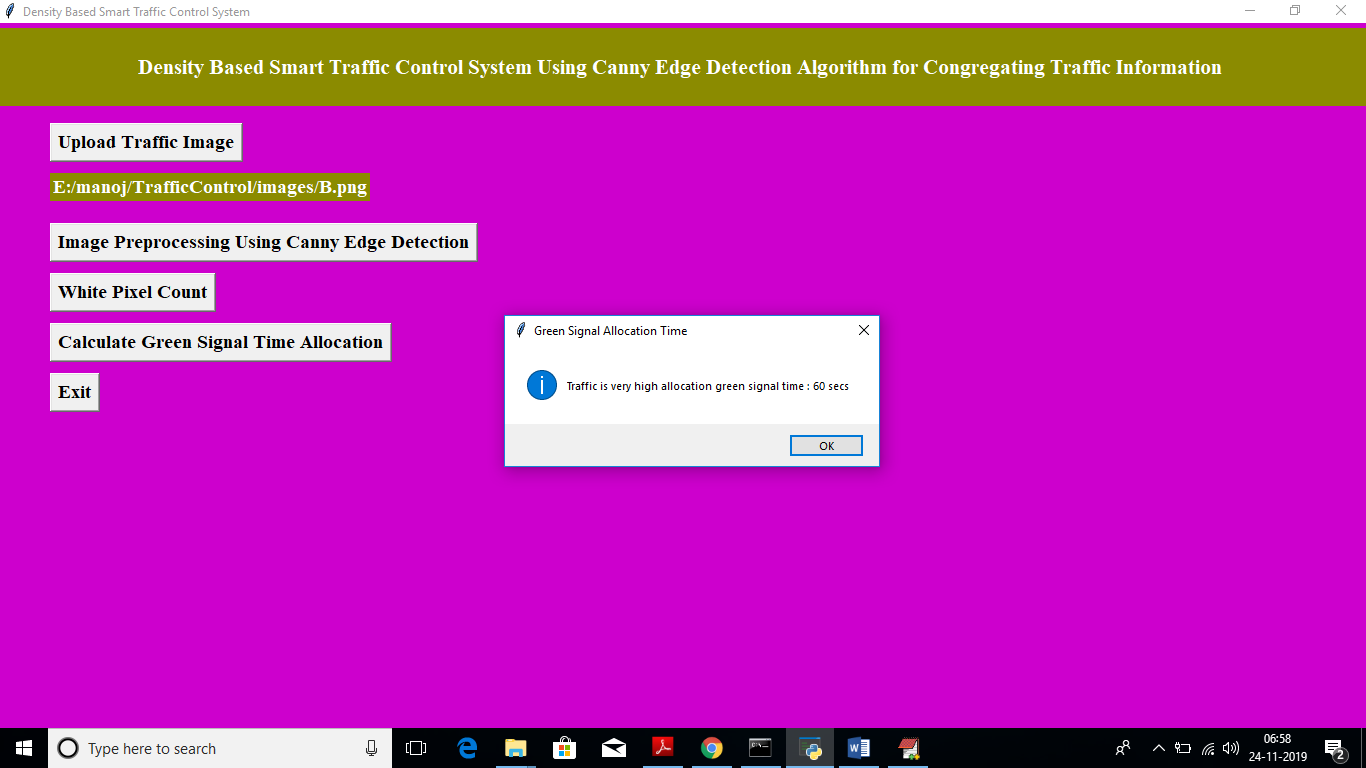
In above screen we got message as input image loaded. Now click on ‘Image Pre-processing Using Canny Edge Detection’ button to apply Gaussian filter and to get canny edges, after clicking button wait for few seconds till you get below screen with edges.



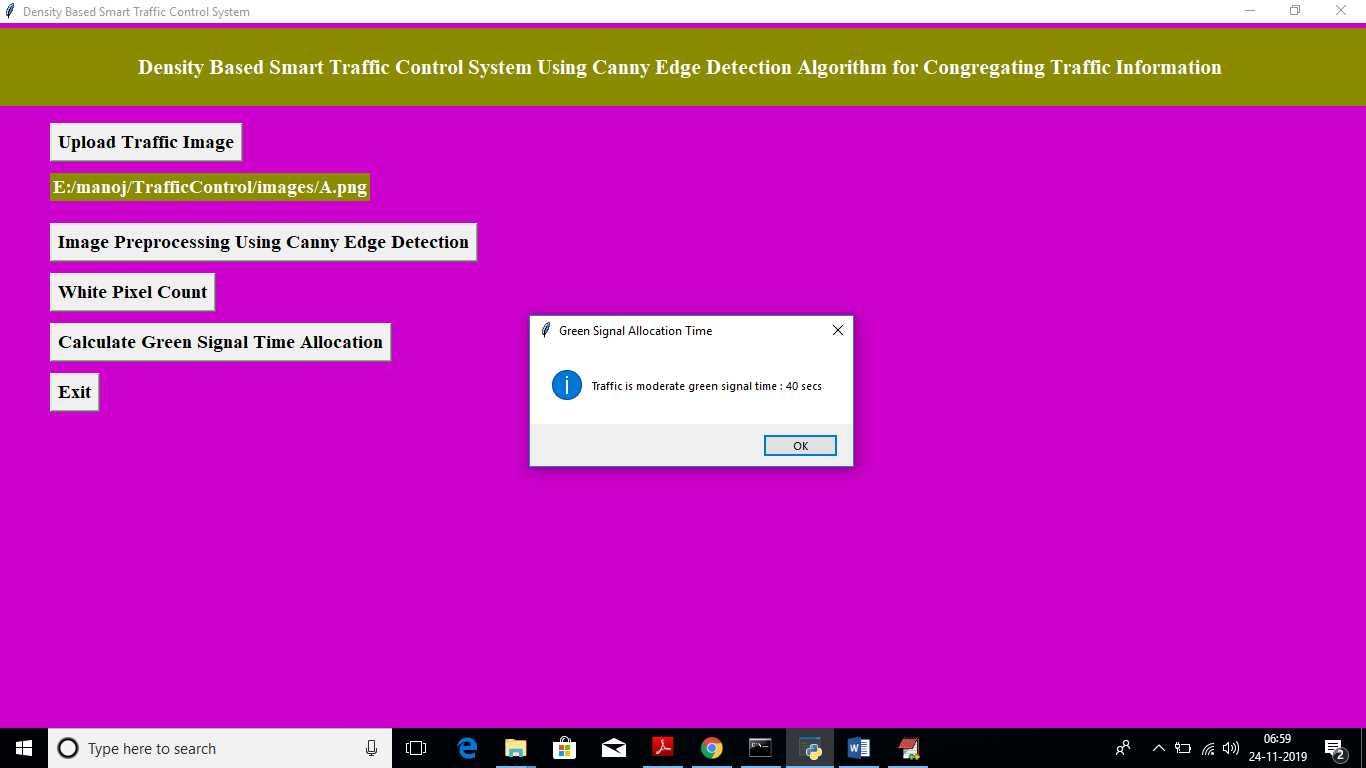
In above screen left side image is the uploaded image and right side is the ‘Reference Image’, Now close this above screen and click on ‘White Pixel count’ button to get white pixels from both images



In above screen dialog box we can see total white pixels found in both sample and reference image. Now click on ‘Calculate Green Signal Time Allocation’ button to get signal time.



For that uploaded image we got message as it contains high traffic and signal time must be 60 seconds. Similarly you can upload any image and get output. Below is the output for image A



Above time for image A

# CHAPTER 6

# TESTING

## Software Testing

Software testing is the process of validating and verifying that a software applicationmeets the technical requirements which are involved in its design and development. It is alsoused to uncover any defects/bugs that exist in the application. It assures the quality of thesoftware. There are many types of testing software viz., manual testing, unit testing, black box testing, performance testing, stress testing, regression testing, white box testing etc. Among theseperformance testing and load testing are the most important one for an android application and nextsections deal with some of these types.

## Black box Testing

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing,and specification-based testing.

## White box Testing

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

## Performance Testings

Performance testing is executed to determine how fast a system or sub-system performsunder a particular workload. It can also serve to validate and verify other quality attributes of thesystem such as scalability, reliability and resource usage.

## Load Testing

Load testing is primarily concerned with testing that can continue to operate underspecific load, whether that is large quantities of data or a large number of users.

## Manual Testing

Manual Testing is the process of manually testing software for defects. Functionality of this application is manually tested to ensure the correctness. Few examples of test case for Manual Testing are discussed later in this chapter.

|  |  |
| --- | --- |
| **Test Case 1** | |
| Test Case Name | Empty login fields testing |
| Description | In the login screen if the username and password fields are empty |
| Output | Login fails showing an alert box asking to enter username and  password. |

**Table 6:1 Test Case for Empty Login Fields**

**Figure 6-1 Test Case for Empty Login Fields**

|  |  |
| --- | --- |
| **Test Case 2** | |
| Test Case Name | Wrong login fields testing |
| Description | A unique username and password are set by administrator. On entering wrong username or password gives. |
| Output | Login fails showing an alert box username or password  incorrect. |

**Table 6:2 Test Case for Wrong Login Fields**

**Figure 6-2 Test Case for Wrong Login Fields**

|  |  |
| --- | --- |
| **Test Case 3** | |
| Test Case Name | User Signup Fails. |
| Description | User signup need to provide all data. |
| Output | Signup Fails and an alert message appears asking to enter valid email and name. |

**Table 6:3 Test Case for Signup fai**

# CHAPTER 7

**RESULTS AND CHALLENGES**

## Results

The current android application is developed using Xml, Java, SQL with Firebase connectivity. It can be used by every individual who are in a need of fulfilling their household services.

At the time of submission of my application was capable of doing the following:

* + Displaying thehome screen with different fragments.
  + Authentication of user by using login screen using Firebase.
  + Home screen to display based on user or service provider.
  + After successful login of user, they can choose the service and book a slot of their particular service provider from the displayed list.
  + Add, update, view, delete the user details.
  + After successful login of service provider, they can view all the bookings that are booked by the users and can attend them one by one.
  + Service provider can also set his preferences to not available, if he’s too busy or many users had already booked him.
  + Service provider has the ability to change their particular radius of location for servicing.
  + He can set up to 10 km radius.
  + Logout and end the session.

## Challenges

* Understanding the connections of SQLite Database is a tricky part and confusing when dealing with multiple tables within a database.
* Making exact orientation API design levels was a difficult task as there are many types of devices like desktop, tablet, mobile with varying screen size and resolutions.
* Implementing synchronization with Firebasewas a challenging task.
* Learning different technologies and frameworks with little guidance.

**CHAPTER 8**

**CONCLUSION**

## Conclusion

In this paper, a smart traffic control system availing image processing as an instrument for measuring the density has been proposed. Besides explaining the limitations of current near obsolete traffic control system, the advantages of proposed traffic control system have been demonstrated. For this purpose, four sample images of different traffic scenario have been attained. Upon completion of edge detection, the similarity between sample images with the reference image has been calculated. Using this similarity, time allocation has been carried out for each individual image in accordance with the time allocation algorithm. In addition, similarity in percentage and time allocation have been illustrated for each of the four sample images using Python programming language. Besides presenting the schematics for the proposed smart traffic control system, all the necessary results have been verified by hardware implementation.

## Scope for future work

## 

The similarity between sample images with the reference image has been calculated. Using this similarity, time allocation has been carried out for each individual image in accordance with the time allocation algorithm. In addition, similarity in percentage and time allocation have been illustrated for each of the four sample images using Python programming language. Besides presenting the schematics for the proposed smart traffic control system, all the necessary results have been verified by hardware implementation.

**9. REFERENCE**

Code snippets for any errors <http://stackoverflow.com/>

Software Testing <http://en.wikipedia.org/wiki/Software_testing>

Manual Testing <http://en.wikipedia.org/wiki/Manual_testing>

Performance Testing <http://en.wikipedia.org/wiki/Software_performance_testing>