CHAPTER 1

1. INTRODUCTION

1.1 Introduction :-

Traffic congestion is one of the major modern-day crisis in every big city in the world. Recent study of World Bank has shown that average vehicle speed has been reduced from 21 km to 7 km per hour in the last 10 years in Dhaka. Inter metropolitan 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, some of the work calculate number of vehicles .These methods have shown promising results in collecting traffic data. However, calculating the number of vehicles may give false results if the intravehicular spacing is very small (two vehicles close to each other may be counted as one) and it may not count rickshaw or auto-rickshaw as vehicles which are the quotidian means of traffic especially in South-Asian countries. And counting number of pixels has disadvantage of counting insubstantial materials as vehicles such as footpath or pedestrians. Some of the work have proposed to allocate time based solely on the density of traffic. But this may be disadvantageous for those who are in lanes that have less frequency of traffic.

1.2 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.

1.3 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.

1.4 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.

1.5 Advantages of proposed system

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 is to take the average of the contribution from each channel: (R+B+C)/3.

CHAPTER 2

2. REQUIREMENT ANALYSIS

2.1 Requirement Analysis

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

2.2 Requirement Specification

2.2.1 Functional requirement

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.

Functional software requirements help you to capture the intended behavior of the system. This behavior may be expressed as functions, services or tasks or which system is required to perform.

2.2.2 Non-Functional requirement

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.

2.3 Computational resource requirements

2.3.1 Hardware requirements :-

✓ Processor	:	Pentium IV or higher
✓ RAM	:	512 MB
✓ Hard Disk	:	40 GB
✓ Mouse	:	Optical Mouse.
✓ Monitor	:	15’ Colour Monitor.

2.3.2 Software requirements :-

✓ Operating System	:	Windows 7 or Above
✓ Compiler / Interpreter	:	Python 3.7
✓ Libraries	:	numpy, scipy, scikit-image & opencv-python,

CHAPTER 3

3. DESIGN

3.1 Design 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.

3.2 Use case Diagram

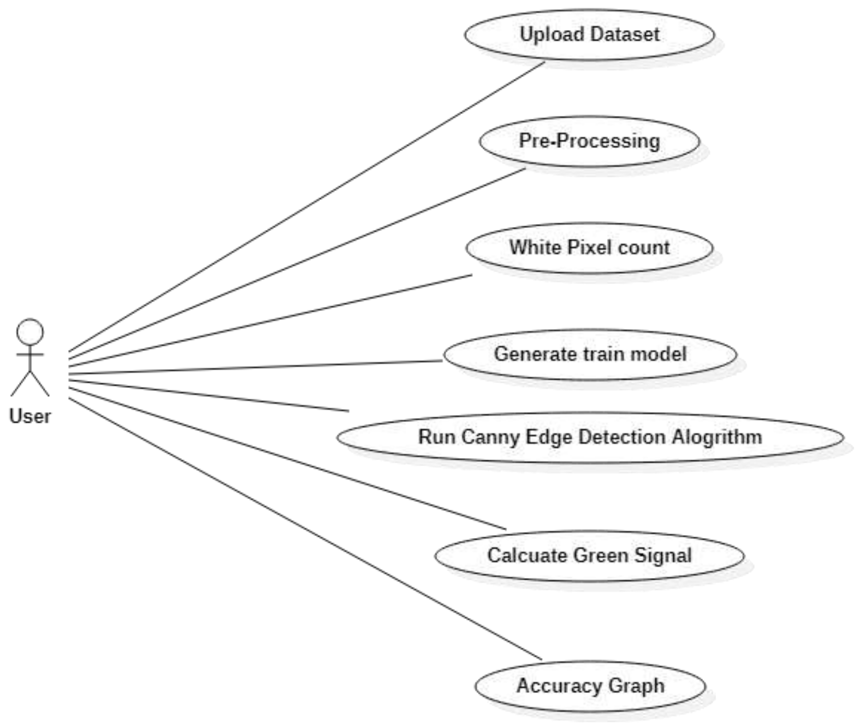


Figure 3.1 Use case Diagram

The above diagram represents the main actor in the project.

- User

3.3 Class Diagram

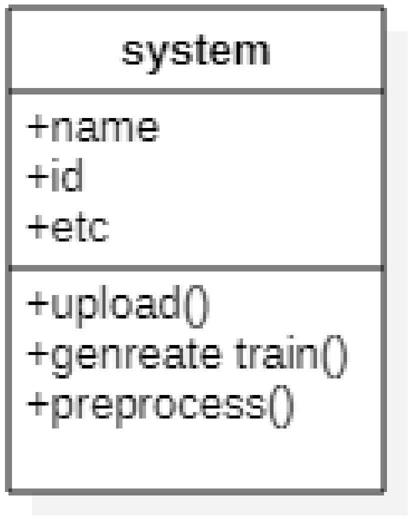


Figure 3.2 class diagram

The above mentioned class diagram represents the system workflow model.

3.4 Sequence Diagram

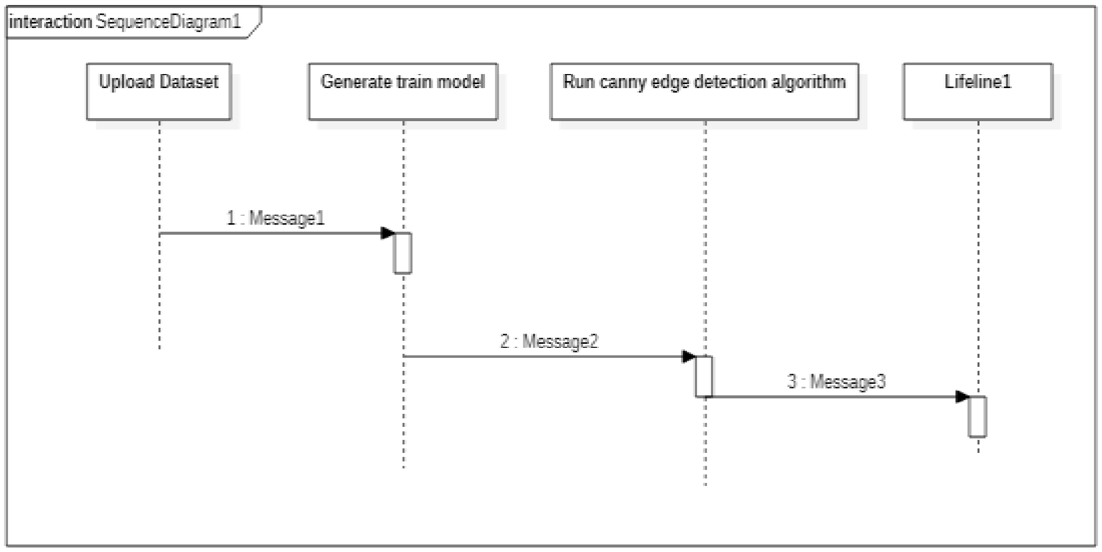


Figure 3.3 Sequence diagram

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

3.5 State Chart Diagram

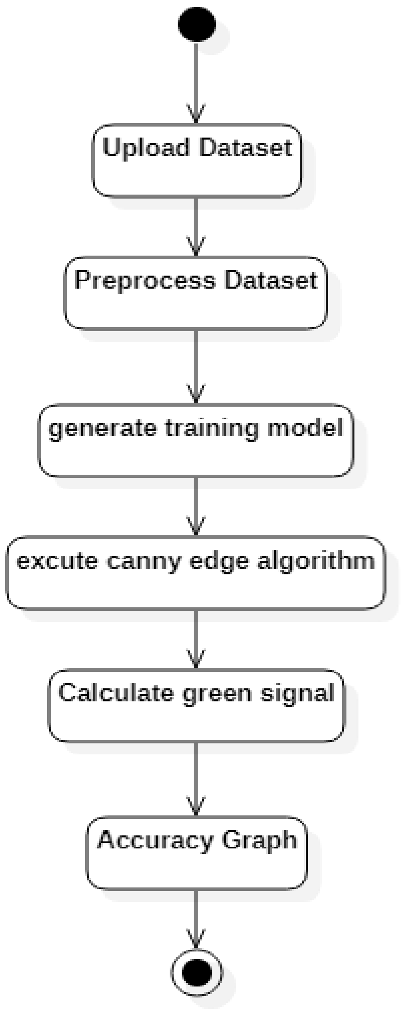


Figure 3.4 State Chart diagram

The above diagram represents the State Chart of flow of action.

CHAPTER 4

4. SYSTEM LOW LEVEL DESIGN

4.1 Modules of the Project

This chapter mainly provides the overview on modules of the application, objectives of the project and a detailed project overview.

4.2 Upload Image Module:

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

4.3 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.

4.4 White Pixel Count Module:

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

CHAPTER 5

5. IMPLEMENTATION

5.1 Sample 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 Sample White Pixels Count : 
"+str(sample_pixels)+"\nTotal Reference 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()
```

CHAPTER 6

6. SCREENSHOTS

6.1 Screenshots.

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

6.1.1 Image Folder

Figure No. 6.1 Image Folder

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

6.1.2 Interface

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

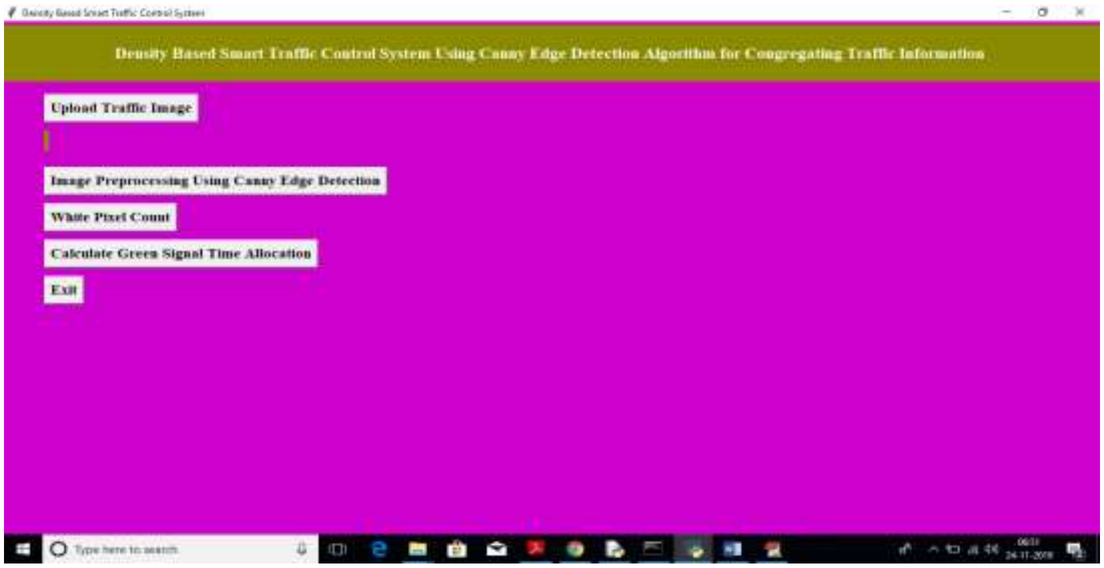

Density Based Smart Traffic Control System
Density Based Smart Traffic Control System Using Canny Edge Detection Algorithm for Congregating Traffic Information
Upload Traffic Image
Image Preprocessing Using Canny Edge Detection
White Pixel Count
Calculate Green Signal Time Allocation
Exit
Type here to search

Figure No. 6.2 Interface

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

6.1.3 Image Selection

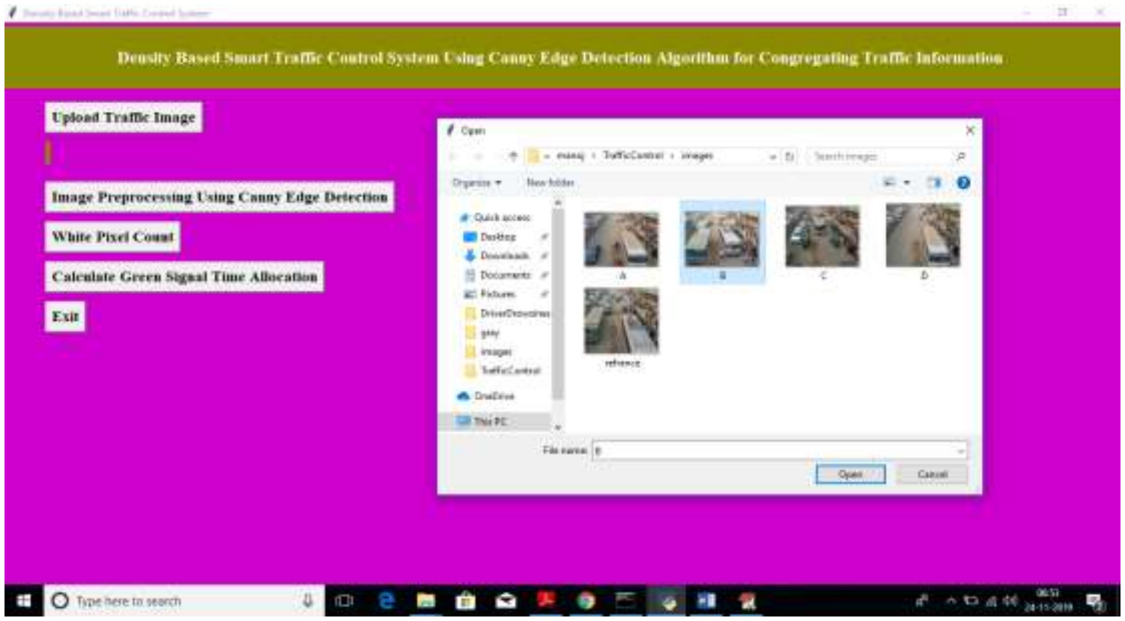

Density Based Smart Traffic Control System Using Canny Edge Detection Algorithm for Congregating Traffic Information
Upload Traffic Image
Image Preprocessing Using Canny Edge Detection
White Pixel Count
Calculate Green Signal Time Allocation
Exit
Open
A
B
C
D
reference
File name: B
Open
Cancel

Figure No. 6.3 Image Selection

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

6.1.4 Image Processing

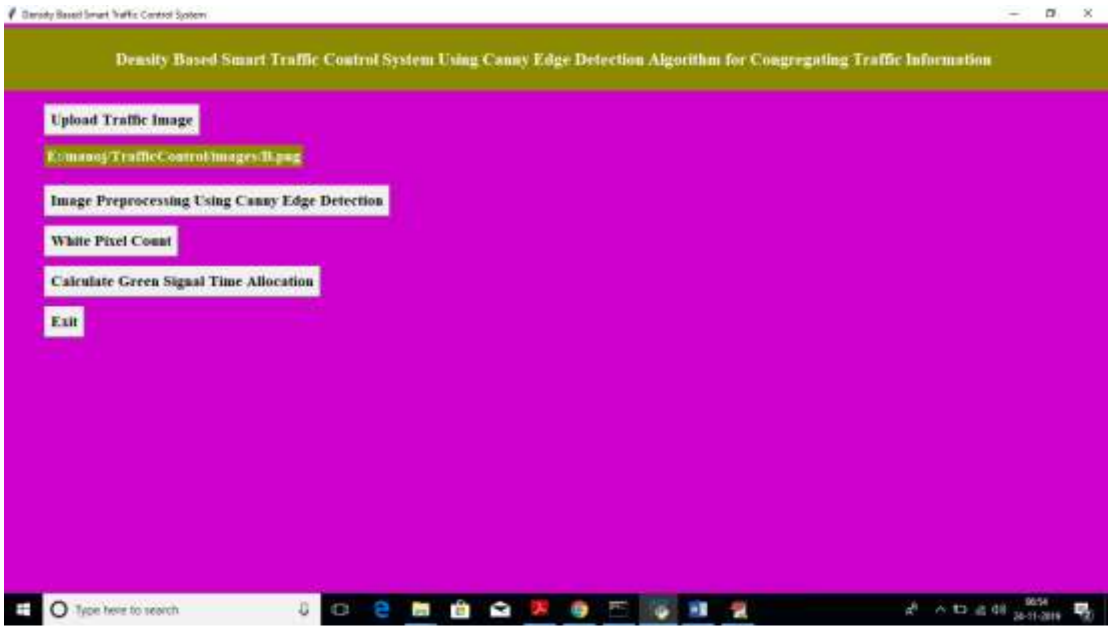

Density Based Smart Traffic Control System
Density Based Smart Traffic Control System Using Canny Edge Detection Algorithm for Congregating Traffic Information
Upload Traffic Image
E:/manoj/TrafficControl/images/B.png
Image Preprocessing Using Canny Edge Detection
White Pixel Count
Calculate Green Signal Time Allocation
Exit
Type here to search

Figure No. 6.4 Image Processing

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

6.1.5 Reference Image

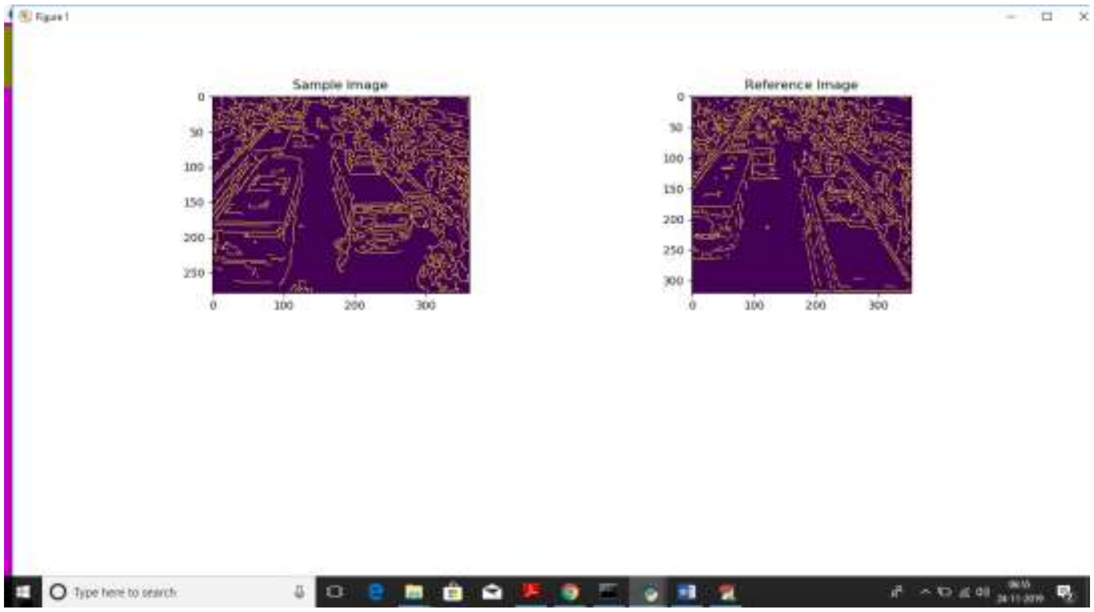
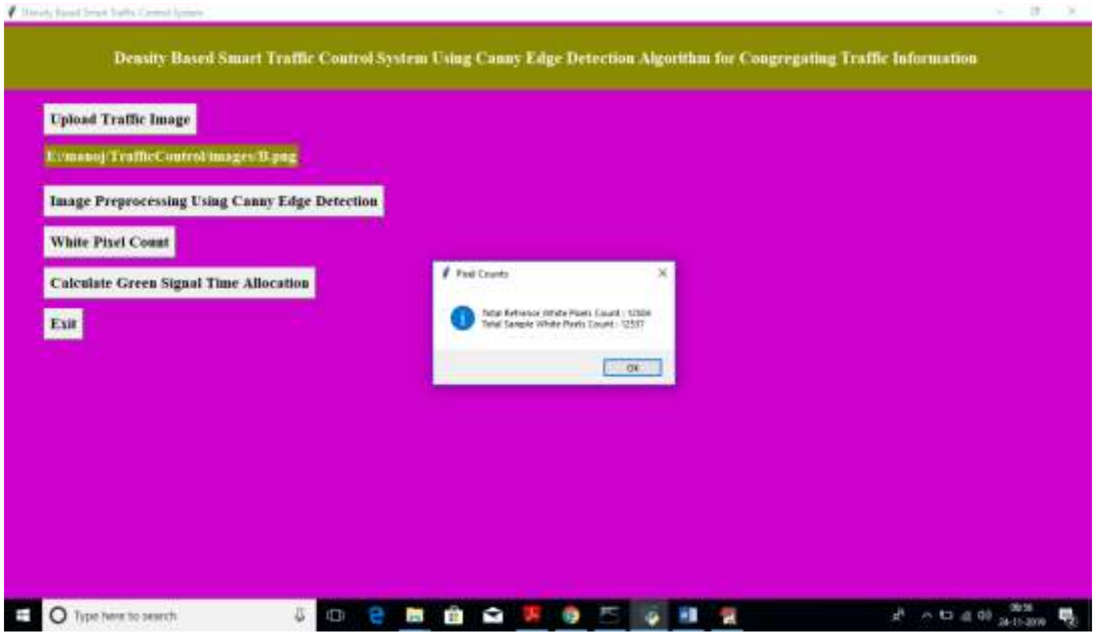


Figure No. 6.5 Reference Image

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

6.1.6 Pixel Counts

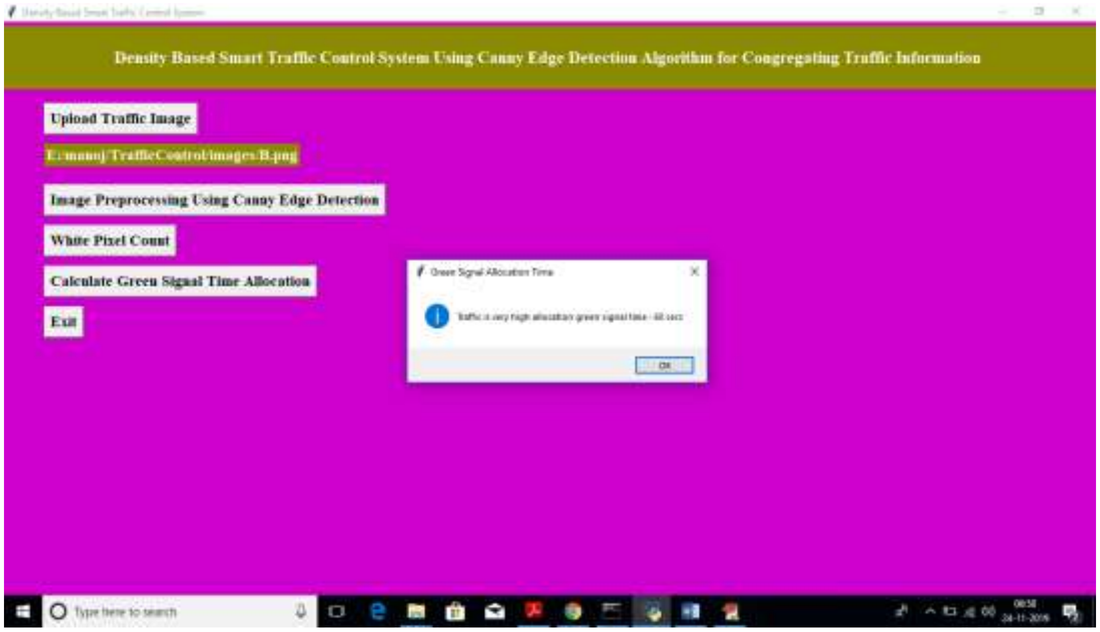

Density Based Smart Traffic Control System Using Canny Edge Detection Algorithm for Congregating Traffic Information
Upload Traffic Image
E:/manoj/TrafficControl/images/B.png
Image Preprocessing Using Canny Edge Detection
White Pixel Count
Calculate Green Signal Time Allocation
Exit
Pixel Counts
OK
Type here to search

Total Reference White Pixels Count : 12504
Total Sample White Pixels Count : 12537

Figure No. 6.6 Pixel Count

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

6.1.7 Green Signal Time


Density Based Smart Traffic Control System Using Canny Edge Detection Algorithm for Congregating Traffic Information

E:/manoj/TrafficControl/images/B.png

Green Signal Allocation Time

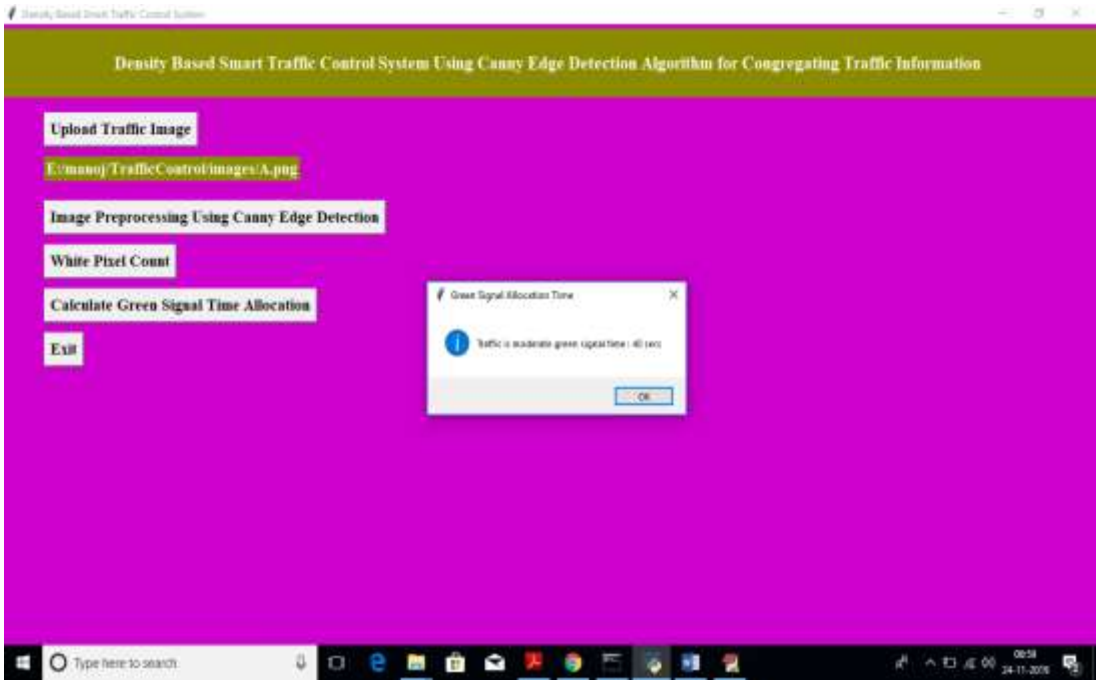
Traffic is very high allocation green signal time : 60 secs

Upload Traffic Image
Image Preprocessing Using Canny Edge Detection
White Pixel Count
Calculate Green Signal Time Allocation
Exit
OK
Type here to search

Figure No. 6.7 Green 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

7.1.8 Image A


Density Based Smart Traffic Control System Using Canny Edge Detection Algorithm for Congregating Traffic Information
Upload Traffic Image
Image Preprocessing Using Canny Edge Detection
White Pixel Count
Calculate Green Signal Time Allocation
Exit
OK
Type here to search

E:/manoj/TrafficControl/images/A.png

Green Signal Allocation Time

Traffic is moderate green signal time : 40 secs

Figure No. 7.8 Image A

Above time for image A

CHAPTER 7

7. TESTING

7.1 Overview of Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.2 Types of Testing

7.2.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.2.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.2.3 Functional testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.3 Unit Testing :-

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.
- Features to be tested
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.4 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.5 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 8

8. CONCLUSION AND FUTURE WORK

8.1 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.

8.2 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.

REFERENCES

[1] Amanjot Kaur, Dr. Mohita Garag , Harpreet Kaur, “Review of traffic management control techniques,” Volume 7, Issue 4, April 2017.

[2] Vishakha S. Thakare, Snehal R.Jadhav, Sananaj G.Sayyed, Poonam Pawar, “Design of smart traffic light controller using embedded system,” Volume 10, Issue 1 (Mar. - Apr. 2013)

[3] Ms.shaili shinde, Prof..Sheetal Jagtap, “Intelligent traffic management system :a review,” IJIRST –International Journal for Innovative Research in Science & Technology| Volume 2 | Issue 09 | February 2016.

[4] R.keerthi, S.hariharagopalan,, “A survey on various traffic management schemes for traffic clearance, stolen vehicle and emergency vehicle,” International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE) ISSN: 0976-1353 Volume 19 Issue 2 – January 2016.

[5] Chandana K K, Dr. S. Meenakshi Sundaram, Cyana D’sa, Meghana NS wamy, Navya K, “Smart traffic management system for congestion

[6] control and warnings using internet of things (IoT),” Chandana K K etal.; Saudi J. Eng. Technol.; Vol-2, Iss-5(May, 2017):192-196

[7] Md.Rokebul Islam, Nafis Ibn Shahid, Dewan Tanzim ul Karim, Abdullah A Mamun, Dr. Md. Khalilur Rhaman, “An efficient algorithm for detecting traffic congestion and a framework for smart traffic control system,” Jam. 31~Feb. 3, 2016.

[8] Tousif Osman, Shahreen Shahjahan Psyche, J. M. Shafi Ferdous, Hasan Zaman, “Intelligent traffic management system for cross section of roads using computer vision”, 2017.

[9] G. Monika, N.Kalpana, Dr.P.Gnanasundari, “An intelligent automatic traffic light controller using embedded system,” Volume 4, Special Issue 4, April 2015.