Density Based Smart Traffic Control System UsingCanny Edge Detection Algorithm for CongregatingTraffic Information

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.

avg = (sample\_pixels/refrence\_pixels) \*100 //in this step we are getting average white pixels

if avg >= 90: //if avg pixels > 90 percent then signal time will be 60 secs. Similarly apply for all

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")

To implement this projects following modules are designed

Upload Image Module: In this module current traffic image will be uploaded to application and then convert colour image 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

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.

Steps to upload image and get edges

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. Percentage matching for different sample images and traffic time allocation for them are demonstrated in section IV. The content of this paper completely serves the purpose of demonstrating the limitations of current traffic control techniques and the solution of this limitations with detailed explanation. Image matching by comparing detected edges is a novel approach to identify the vehicular density with propitious accuracy. As far as we know, matching images by comparing detected edges has not been used before for smart traffic control application.

Canny Edge Detection Algorithm Description

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:

1. Pre-processing
2. Calculating gradients
3. Non-maximum suppression
4. 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:

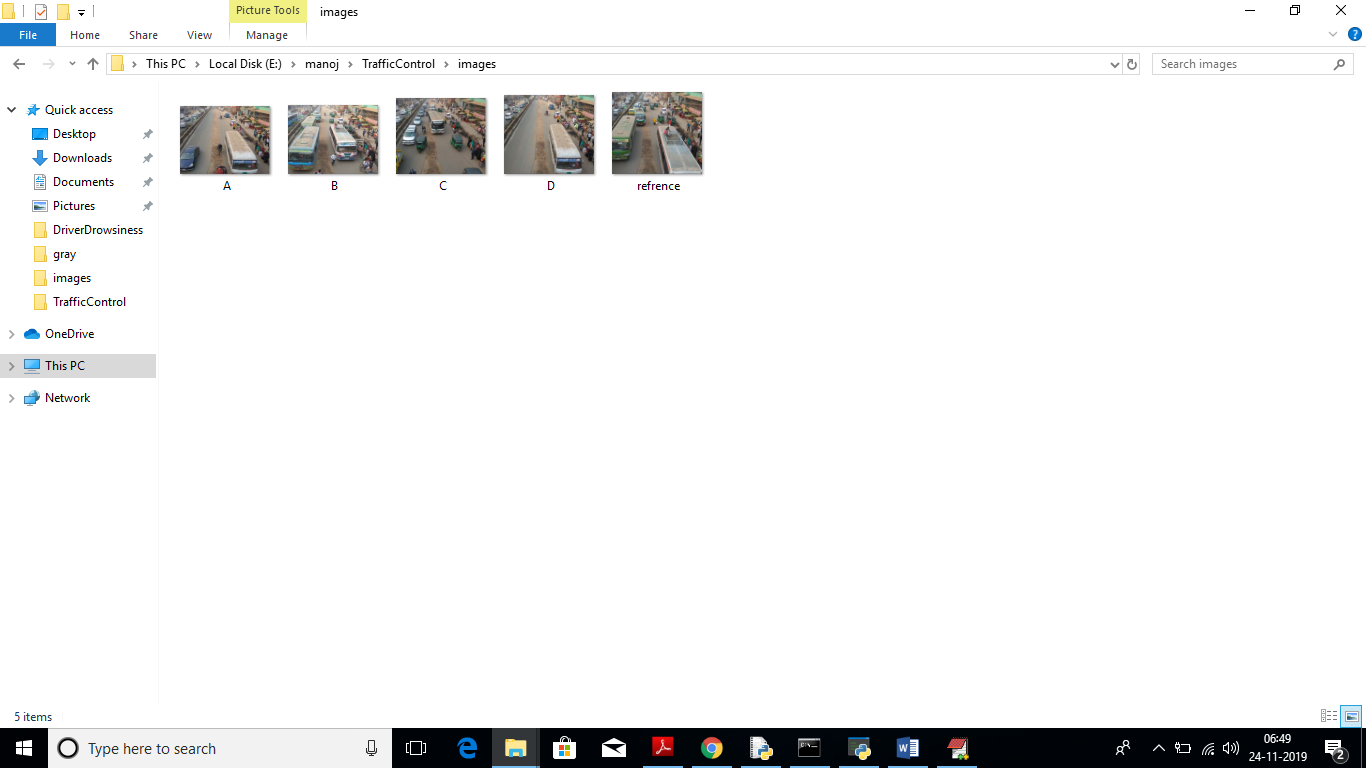
Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible.

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.

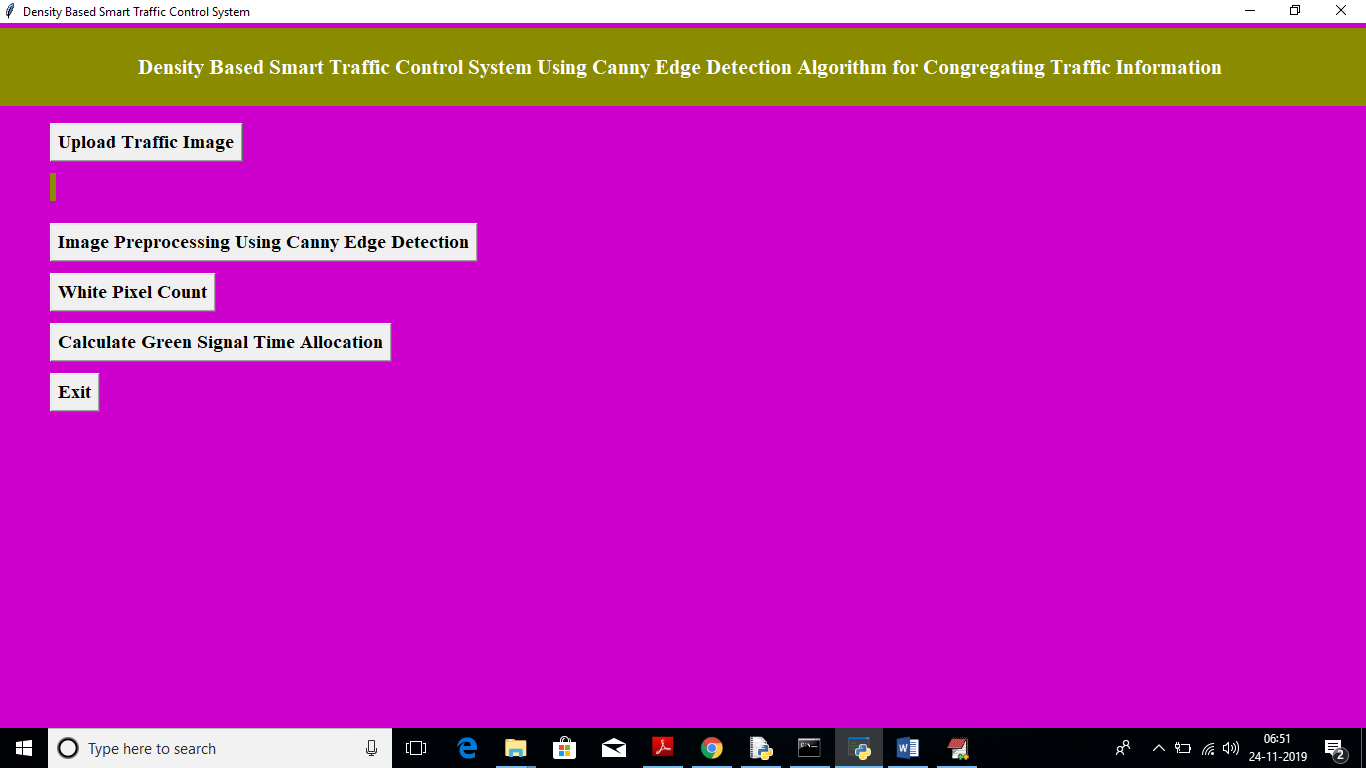
To implement this project we are using 4 input images given in paper and on reference image. Below are the images screen shots saved inside images folder



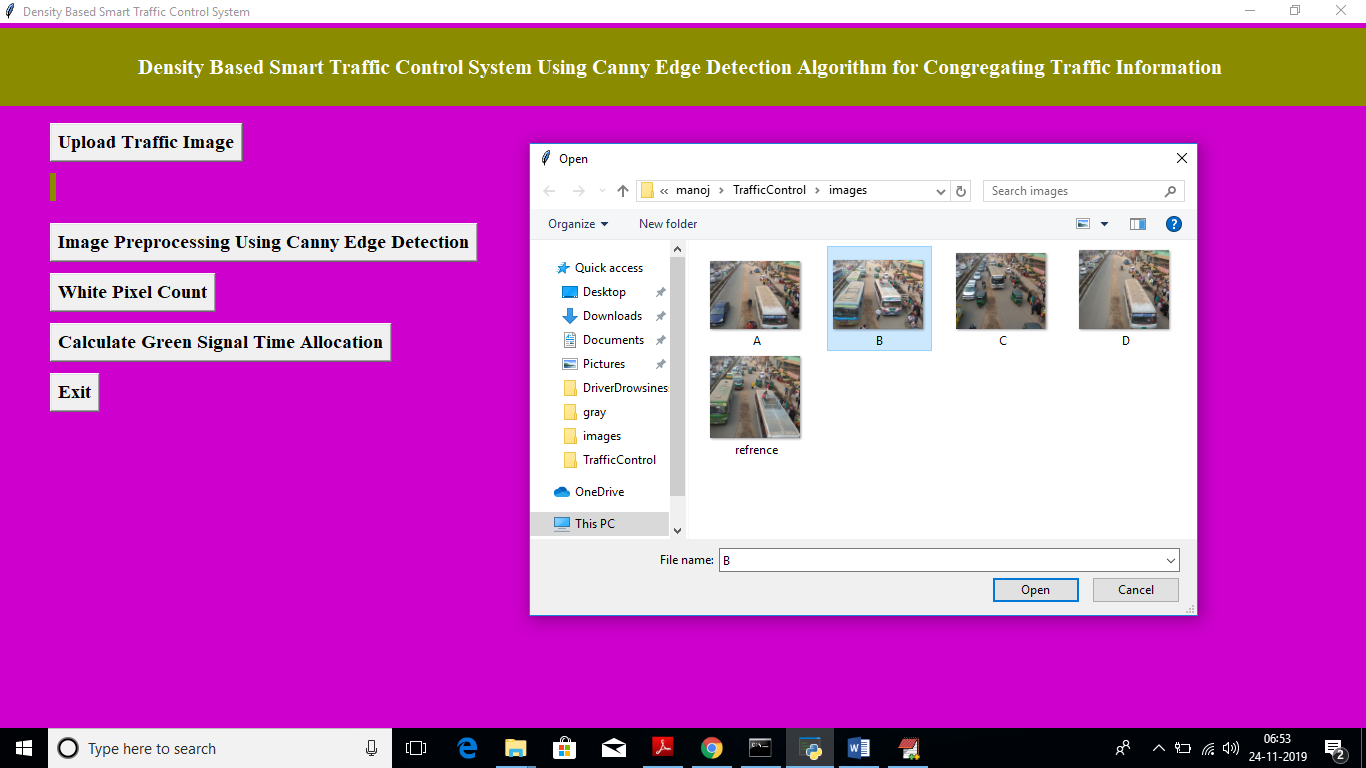
We can upload above 4 images to application to calculate traffic signal time.

Screen shots

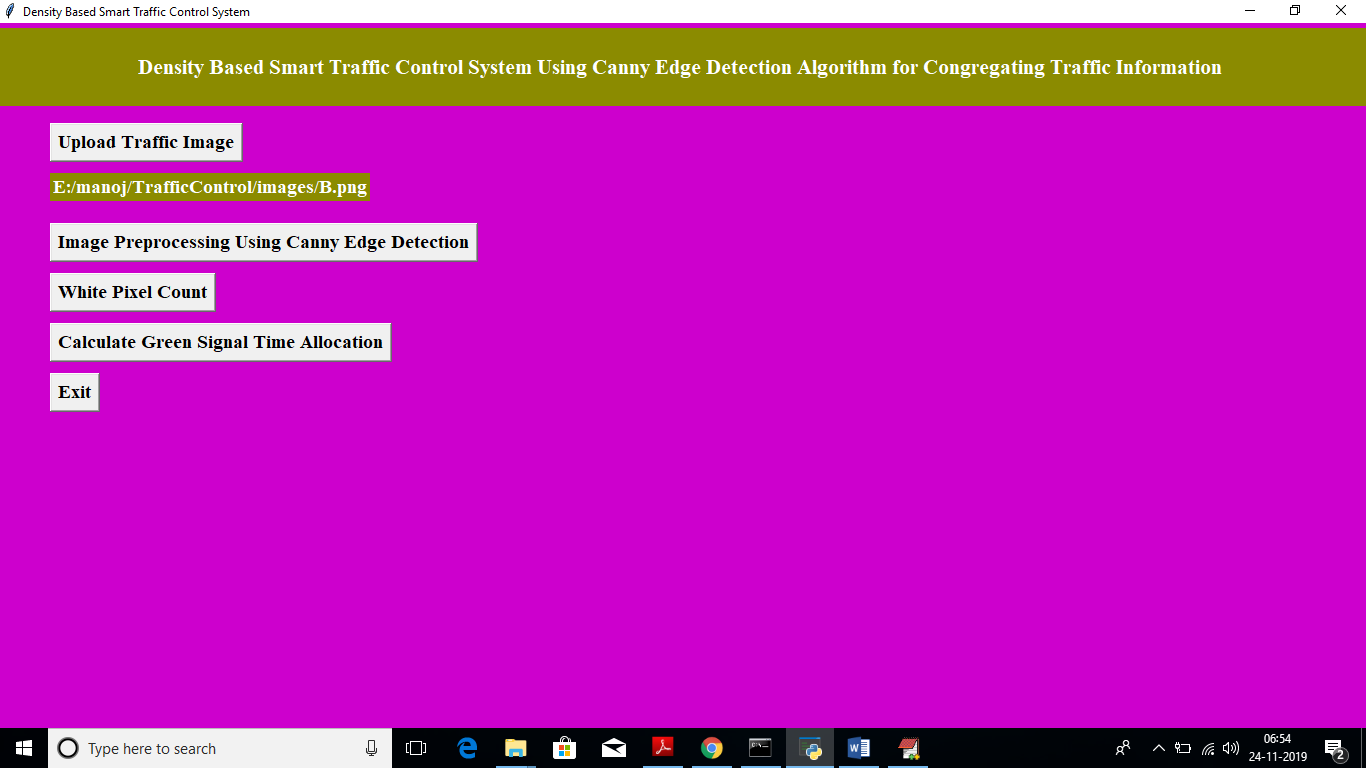
To run this project double click on ‘run.bat’ file to get below screen



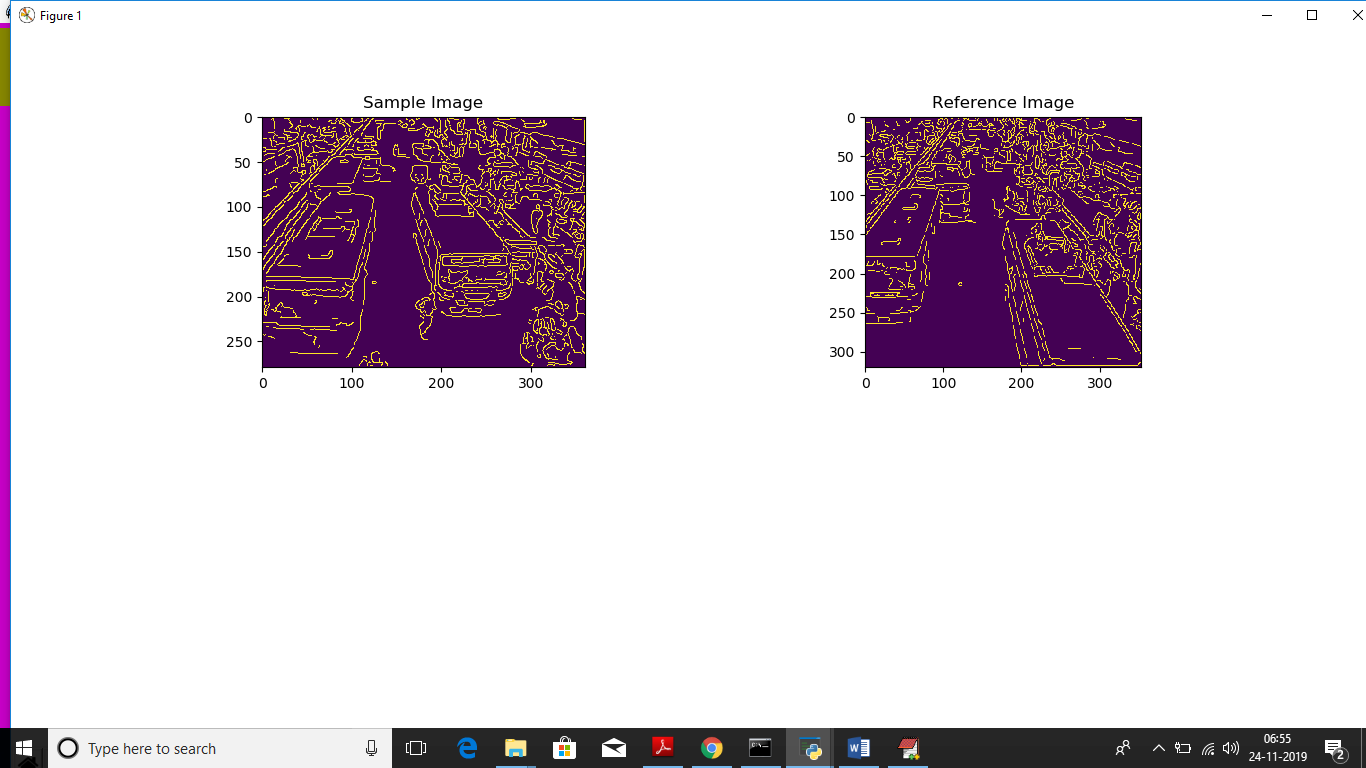
In above screen click on ‘Upload Traffic Image’ button to upload image.



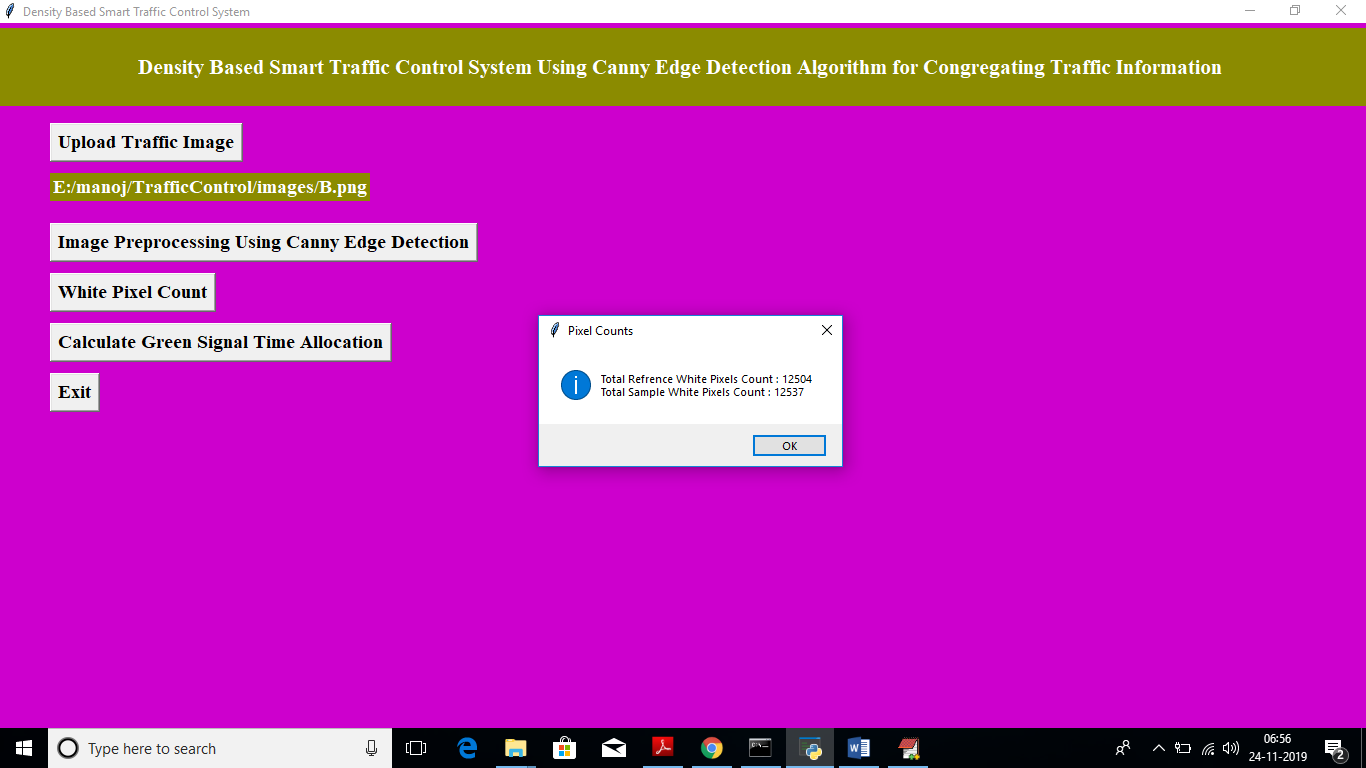
In above screen I am uploading image B and now click on ‘Open’ button to load image



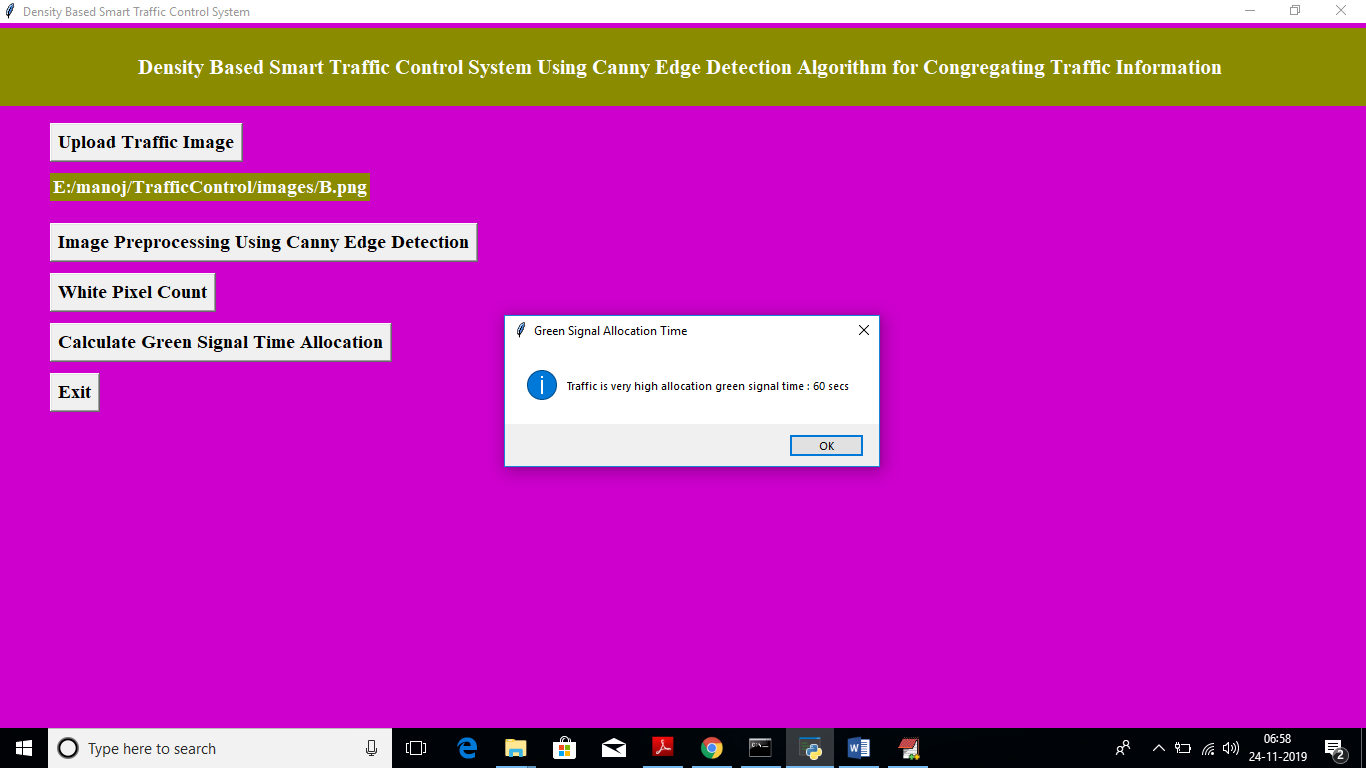
In above screen we got message as input image loaded. Now click on ‘Image Pre-processing Using Canny Edge Detection’ button to apply Gaussian filter and to get canny edges, after clicking button wait for few seconds till you get below screen with edges



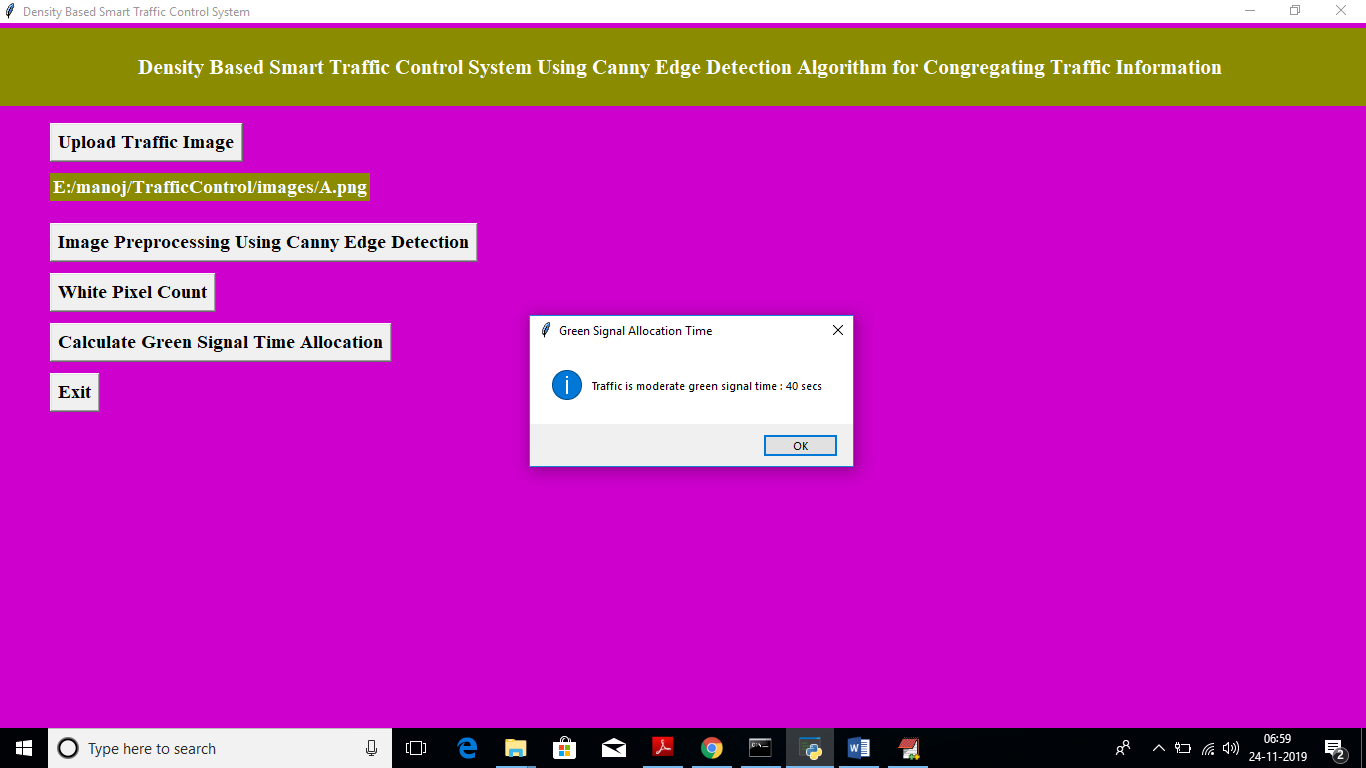
In above screen left side image is the uploaded image and right side is the ‘Reference Image’, Now close this above screen and click on ‘White Pixel count’ button to get white pixels from both images



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