



**VIT<sup>®</sup>**

**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**November 2018**

# **Smart Traffic Light System**

## **A Project Report**

**Under the Guidance of,  
Prof. A. Srivani**

**By**

**– Name of the Student**

17BCE0939 – Sahil Gami

17BCE2305 – Sreeparna Deb

17BCE2223 – Tejas Helwatkar

17BCE2296 – Yash Asawa

## **DECLARATION BY THE CANDIDATES**

We hereby declare that the project report entitled **“Smart Traffic Light System”** submitted by us to Vellore Institute of Technology, Vellore in partial fulfilment of the requirement for the award of the degree of **B. Tech (CSE)** is a record of J- component of project work carried out by us under the guidance of **Prof. A. Srivani**. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore Institute of Technology, Vellore.

Date :

Signature of the faculty

Signature of the Candidate

# **TABLE OF CONTENTS**

## **1. Introduction**

- 1.1 Abstract
- 1.2 Background

## **2. Overview and Planning**

- 2.1 Proposed Work
- 2.2 Hardware Requirements
- 2.3 Software Requirements

## **3. Literature Survey and Review**

- 3.1 Literature Summary

## **4. Methodology**

- 4.1 Method Used
- 4.2 Applications

## **5. System Implementation**

- 5.1 Code
- 5.2 Results and discussion

## **6. Conclusion**

- 6.1 Conclusion
- 6.2 Future Work

## **7. References**

# 1. Introduction

## 1.1 Abstract

The conventional Traffic light system is time based, meaning it stays green in a lane for particular amount of time for the lane, at that moment other lane's signals are red. After that particular time interval, green light turns to red and another lane's light turns to green. No matter the traffic strength.

This conventional system is organised based assuming that on all the lanes, the traffic is equivalent on all four lanes, a scenario which is hardly possible. This system can be very annoying if there is very less or no vehicle on a lane and still its light is green based on the conventional system. Whereas if there is huge traffic on the other lane, still it will have to wait for turn of its lane's light to turn green.

This project aims to make a smart traffic lights control program which can identify traffic strength on each lane and accordingly set the timers of lane. The basic aim of the program will be to minimize traffic jam on cross roads based on traffic intensity of lanes.

## 1.2 Background

In modern life we have to face with many problems one of which is traffic congestion becoming more serious day after day. It is said that the high tome of vehicles, the scanty infrastructure and the irrational distribution of the development are main reasons for augmented traffic jam. The major cause leading to traffic jam is the high number of vehicles which was caused by the population and the development of economy.

Many techniques have been developed in Image Processing during the last four to five decades. Most of the methods are developed for enhancing images obtained from unmanned space probes, spacecrafts and military reconnaissance flights. Image Processing systems are becoming widely popular due to easy availability of powerful personnel computers, large memory devices, graphics softwares and many more.

Image processing involves issues related to image representation, compression techniques and various complex operations, which can be carried out on the image data. The operations that come under image processing are image enhancement operations such as sharpening, blurring, brightening, edge enhancement. Traffic density of lanes is calculated using image processing which is done of images of lanes that are captured using digital camera. We have chosen image processing for calculation of traffic density as cameras are very much cheaper than other devises such as sensors.

Making use of the above-mentioned virtues of image processing we propose a technique that can be used for traffic control.

## **2. Overview and Planning**

### **2.1 Proposed Work**

We propose a system for controlling the traffic light by image processing. The vehicles are detected by the system through images instead of using electronic sensors embedded in the pavement. A camera will be placed alongside the traffic light. It will capture image sequences. Image processing is a better technique to control the state change of the traffic light. It shows that it can decrease the traffic congestion and avoids the time being wasted by a green light on an empty road. It is also more reliable in estimating vehicle presence because it uses actual traffic images. It visualizes the practicality, so it functions much better than those systems that rely on the detection of the vehicles' metal content.

### **2.2 Hardware Requirements**

A camera fixed on a rotating motor which rotates 90 degrees after clicking an image of a lane and transmits to computer for logical (However due to lack of apparatus, we will be using mobile phone to click pictures, or already taken images will be used to illustrate the system).

### **2.3 Software Requirements**

MATLAB (with Image processing toolbox) will be used to process images and design a logical code for the traffic light.

### **3. Literature Survey and Review**

#### **3.1 Literature Summary**

While going through all the research papers for different projects based on image processing which used MATLAB, many common things are taken into consideration like filtering an image off its noise, is mainly done by median filter as it does not render the clarity of an image as well as it does not affect the sharpness of the image.

Sobel edge detection was mentioned in mostly every paper and on the internet while searching for many different edge detection techniques, and so it was taken into consideration for the use in the mentioned project.

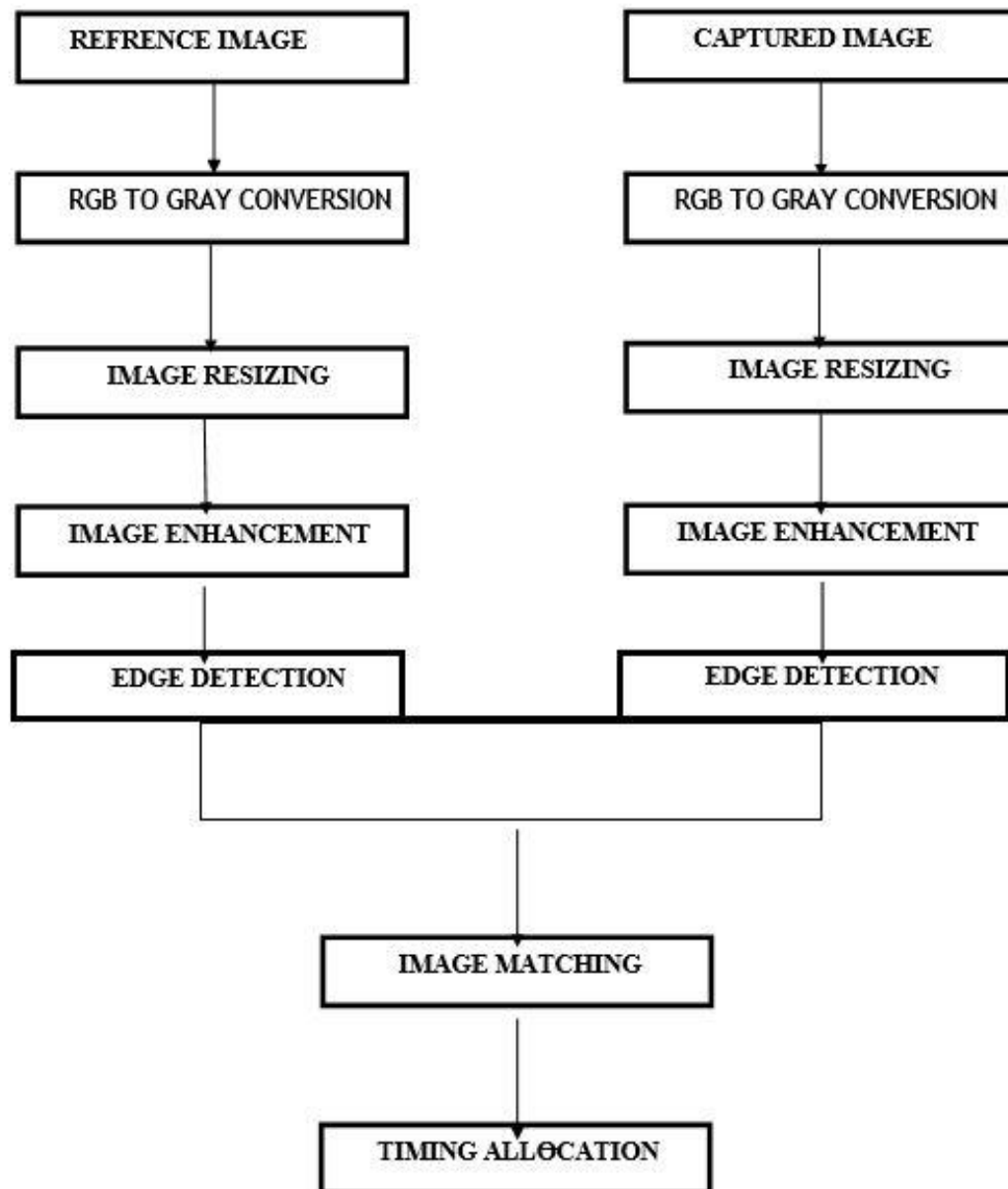
Many different techniques of performing the given project were discussed and one of the best suitable is used out of all the previously discussed methods, like calculating change in the image, calculating change in the video frame, calculating the number of cars or particular vehicles, etc. pros and cons of each were noted and then one final technique to calculate the change in image with respect to reference image was taken into consideration and further worked on with calculating percentage of vehicles in the image, which did not work well as expected.

After considering the need availability and capability of the current system and budget, one final solution was discussed and worked upon, which still requires some more modifications, but as of now, the current project is a main idea what it can bring to daily use of the project idea properly, and the amount of time it can save SMARTLY.

Also, the effort is made to detect any emergency vehicles on any particular lane, but it is not efficiently working as of now, and requires some more modification, articles related to OCR were collected and studied to make a better OCR function, however it is not achieved yet, it can be further worked upon and completed after some more further efforts.

## 4. Methodology

### 4.1 Method Used



#### Image Acquisition

Generally an image is a two-dimensional function  $f(x,y)$  (here  $x$  and  $y$  are plane coordinates). The amplitude of image at any point say  $f$  is called intensity of the image. It is also called the gray level of image at that point. We need to convert these  $x$  and  $y$  values to finite discrete values to form a digital image.

The input image is a fundus taken from stare data base and drive data base. The image of the retina is taken for processing and to check the condition of the person.

We need to convert the analog image to digital image to process it through digital computer. Each digital image composed of a finite elements and each finite element is called a pixel.

### Formation of Image

We have some conditions for forming an image  $f(x,y)$  as values of image are proportional to energy radiated by a physical source. So  $f(x,y)$  must be nonzero and finite.

i.e.  $0 < f(x,y) < \infty$ .

### Image Pre-Processing

#### Image Resizing/Scaling:

Image scaling occurs in all digital photos at some stage whether this be in Bayer demosaicing or in photo enlargement. It happens anytime you resize your image from one pixel grid to another. Image resizing is necessary when you need to increase or decrease the total number of pixels. Even if the same image resize is performed, the result can vary significantly depending on the algorithm.

Images are resized because of number of reasons but one of them is very important in our project. Every camera has its resolution, so when a system is designed for some camera specifications it will not run correctly for any other camera depending on specification similarities. so it is necessary to make the resolution constant for the application and hence perform image resizing.

### RGB to GRAY Conversion

Humans perceive colour through wavelength-sensitive sensory cells called cones. There are three different varieties of cones, each has a different sensitivity to electromagnetic radiation (light) of different wavelength. One cone is mainly sensitive to green light, one to red light, and one to blue light.

By emitting a restricted combination of these three colours (red, green and blue), and hence stimulate the three types of cones at will, we are able to generate almost any detectable colour.



This is the reason behind why colour images are often stored as three separate image matrices; one storing the amount of red (R) in each pixel, one the amount of green (G) and one the amount of blue (B)

We call such colour images as stored in an RGB format. In grayscale images, however, we do not differentiate how much we emit of different colours, we emit the same amount in every channel.

We will be able to differentiate the total amount of emitted light for each pixel; little light gives dark pixels and much light is perceived as bright pixels.

When converting an RGB image to grayscale, we have 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:  $(R+B+C)/3$ . However, since the perceived brightness is often dominated by the green component, a different, more "human-oriented", method is to consider a weighted average, e.g.:  $0.3R + 0.59G + 0.11B$ .

## Image Enhancement

Image enhancement is the process of adjusting digital images so that the results are more suitable for display or further analysis. For example, we can eliminate noise, which will make it more easier to identify the key characteristics.

In poor contrast images, the adjacent characters merge during binarization. We have to reduce the spread of the characters before applying a threshold to the word image. Hence, we introduce “**POWER- LAW TRANSFORMATION**” which increases the contrast of the characters and helps in better segmentation. The basic form of power-law transformation is

$$s = cr^\gamma,$$

where  $r$  and  $s$  are the input and output intensities, respectively;  $c$  and  $\gamma$  are positive constants. A variety of devices used for image capture, printing, and display respond according to a power- law.

By convention, the exponent in the power-law equation is referred to as gamma. Hence, the process used to correct these power-law response phenomena is called gamma correction. Gamma correction is important, if displaying an image accurately on a

computer screen is of concern. In our experimentation,  $\gamma$  is varied in the range of 1 to 5. If  $c$  is not equal to '1', then the dynamic range of the pixel values will be significantly affected by scaling.

Thus, to avoid another stage of rescaling after power-law transformation, we fix the value of  $c = 1$ . With  $\gamma = 1$ , if the power-law transformed image is passed through binarization, there will be no change in the result compared to simple binarization. When  $\gamma > 1$ , there will be a change in the histogram plot, since there is an increase of samples in the bins towards the gray value of zero. Gamma correction is important if displaying an image accurately on computer screen is of concern.

## Edge Detection

Edge detection is the name for a set of mathematical methods which aim at identifying points in a digital image at which the image brightness changes sharply or, more technically, has discontinuities or noise. The points at which image brightness alters sharply are typically organized into a set of curved line segments termed edges.

The same problem of detecting discontinuities in 1D signal is known as step detection and the problem of finding signal discontinuities over time is known as change detection. Edge detection is a basic tool in image processing, machine vision and computer envisage, particularly in the areas of feature reveal and feature extraction.

## Edge detection techniques

Different colours has different brightness values of particular colour. Green image has more bright than red and blue image or blue image is blurred image and red image is the high noise image.

Following are list of various edge-detection methods: -

- Sobel Edge Detection Technique
- Perwitt EdgeDetection
- Roberts Edge Detection Technique
- Zerocross Threshold Edge Detection Technique
- Canny Edge Detection Technique

In our project we use “CANNY EDGE DETECTION TECHNIQUE” because of its various advantages over other edge detection techniques.

## Canny Edge Detection

The Canny Edge Detector is one of the most commonly used image processing tools detecting edges in a very robust manner. It is a multi-step process, which can be implemented on the GPU as a sequence of filters. Canny edge detection technique is based on three basic objectives.

Low error rate:-

All edges should be found, and there should be no spurious responses. That is, the edges must be as close as possible to the true edges.

Edge point should be well localized:

The edges located must be as close as possible to the true edges. That is, the distance between a point marked as an edge by the detector and the centre of the true edge should be minimum.

Single edge point response:-

The detector should return only one point for each true edge point. That is, the number of local maxima around the true edge should be minimum. This means that the detector should not identify multiple edge pixels where only a single edge point exist.

The essence of Canny’s work was in expressing the preceding three criteria mathematically and then attempting to find optimal solution to these formulations, in general, it is difficult to find a close form solution that satisfies all the preceding objectives. However, using numerical optimization with 1-D step edges corrupted by additive white Gaussian noise led to the conclusion that a good approximation to the optimal step edge detector is the first derivative of Gaussian:

$$\frac{d}{dx} e^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \quad (1)$$

Generalizing this result to 2-D involves recognizing that the 1-D approach still applies in the direction of the edge normal. Because the direction of the normal is unknown beforehand, this would require applying the 1-D edge detector in all possible directions.

This task can be approximated by first smoothing the image with circular 2-D Gaussian function, computing the gradient of the result, and then using the gradient magnitude and direction to estimate edge strength and direction at every point.

Let  $f(x,y)$  denote the input image and  $G(x,y)$  denote the Gaussian function:

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2)$$

We form a smoothed image,  $f_s(x, y)$ , by convolving  $G$  and  $f$ :

$$f_s(x,y) = G(x,y) * f(x,y) \quad (3)$$

This operation is followed by computing the gradient and direction (angle)

$$M(x,y) = \sqrt{g_x^2 + g_y^2} \quad (4)$$

And

$$\theta(x,y) = \tan^{-1} \frac{g_y}{g_x} \quad (5)$$

$$\text{with } g_x = \frac{\partial f_s}{\partial x} \text{ and } g_y = \frac{\partial f_s}{\partial y}.$$

Equation (2) is implemented using an  $n \times n$  Gaussian mask. Keep in mind that  $M(x, y)$  and  $\theta(x, y)$  are arrays of the same size as the image from which they are computed. Because it is generated using the gradient  $M(x, y)$  typically contains wide ridges around local maxima.

The next step is to thin those ridges. One approach is to use non maxima suppression. This can be done in several ways, but the essence of the approach is to specify a no. of discrete orientations of edge normal (gradient vector). For example, in 3x3 region we can define four orientation for an edge passing through the centre point of the region: horizontal, vertical, +45° and -45°.

The final operation is to threshold  $g_N(x, y)$  to reduce false edge point. We do it by using a single threshold, in which all value below the threshold were set to 0. If we set the threshold too low, there will still be some false edge (called false positives). If the threshold is set too high, then actual valid edge points will be eliminated (false negatives). Canny's algorithm attempts to improve on this situation by using hysteresis threshold. We use two threshold,

A low threshold,  $T_L$ , and a high threshold,  $T_H$ . Canny suggested that the ratio of high to low threshold should be two or three to one

We visualize the thresholding operation as creating two additional images

$$g_{NH}(x,y) = g_N(x,y) \geq T_H \quad (6)$$

And

$$g_{NL}(x,y) = g_N(x,y) \geq T_L \quad (7)$$

Where, initially, both  $g_{NH}(x,y)$  and  $g_{NL}(x,y)$  are set to 0. After thresholding,  $g_{NH}(x,y)$  will have fewer nonzero pixels than  $g_{NL}(x,y)$  in general, but all the nonzero pixels in  $g_{NH}(x,y)$  will be contained in  $g_{NL}(x,y)$  because the latter image is formed with lower threshold. We eliminate from  $g_{NL}(x,y)$  all the nonzero from  $g_{NH}(x,y)$  by letting

$$g_{NL}(x,y) = g_{NL}(x,y) - g_{NH}(x,y) \quad (8)$$

The nonzero pixels in  $g_{NL}(x,y)$  and  $g_{NH}(x,y)$  may be viewed as being “strong”. And “weak” edge pixels, respectively.

After the thresholding operations, all strong pixels in  $g_{NH}(x, y)$  are assumed to be valid edge pixels and are so marked immediately. Depending on the value of  $T_H$ , the edges in  $g_{NH}(x, y)$  typically have gaps.

Longer edges are formed using the following procedure

- a Locate the next unvisited pixel  $p$  in  $g_{NH}(x, y)$ .
- b Mark as valid edge pixels all the weak pixels in  $g_{NL}(x, y)$  that are connected to  $p$  using say 8 connectivity.
- c If all nonzero pixels in  $g_{NH}(x, y)$  have been visited go to step d. else return to step a.
- d Set to zero all pixels in  $g_{NL}(x, y)$  that were not marked as valid edge pixels.

At the end of this procedure, the final image output by the Canny is formed by appending to  $g_{NH}(x, y)$  all the nonzero pixels from  $g_{NL}(x, y)$

We use two additional images,  $g_{NH}(x, y)$  and  $g_{NL}(x, y)$ , to simplify the discussion. In practice, hysteresis threshold can be implemented directly during nonmaxima suppression, and thresholding can be implemented directly on  $g_{NL}(x, y)$  by forming a list of strong pixels and the weak pixels connected to them.

Summarizing, the Canny edge detection algorithm consist of the following basic steps;

- i. Smooth the input image with Gaussian filter.
- ii. Compute the gradient magnitude and angle images.
- iii. Apply nonmaxima suppression to the gradient magnitude image.
- iv. Use double thresholding and connectivity analysis to detect and link edges.

## Image Matching

Recognition techniques based on matching represent each class by a prototype pattern vector. An unknown pattern is assigned to the class to which is closest in terms of predefined metric. The simplest approach is the minimum distance classifier, which, as its name implies, computes the (Euclidean) distance between the unknown and each of the prototype vectors. It chooses the smallest distance to make decision. There is another approach based on correlation, which can be formulated directly in terms of images and is quite intuitive.

We have used a totally different approach for image matching. Comparing a reference image with the real time image pixel by pixel. Though there are some disadvantages related to pixel based matching but it is one of the best techniques for the algorithm which is used in the project for decision making.

Real image is stored in matrix in memory and the real time image is also converted in the desired matrix. For images to be same their pixel values in matrix must be same. This is the simplest fact used in pixel matching. If there is any mismatch in pixel value it adds on to the counter used to calculate number of pixel mismatches. Finally, percentage of matching is expressed as

$$\%match = \frac{\text{No.of pixels matched successfully}}{\text{total no.of pixel}}$$

## 4.2 Applications

Robotics vision, classification of medical images, fingerprints for diseases like tumors, identifying objects immersed in images, self-driving vehicles, Where thresholding does not always work, edge detection ay just do the work fine.

## 5. System Implementation

### Implementation Algorithm

The block diagram of the project was discussed in previous chapter. The algorithm behind the block diagram consists of following steps

1. We have a reference image and the image to be matched is continuously captured using a camera that is installed at the junction.
2. The images are pre-processed in two steps as follows
  - a. Images are rescaled to 300x300 pixels.
  - b. Then the above rescaled images are converted from RGB to gray.
3. Edge detection of pre-processed images is carried out using Canny edge detection technique.
4. The output images of previous step are matched using pixel to pixel matching technique.
5. After matching, the timing allocation is done by comparing the percentage of change of each lane and the resultant sequence is stored in a list



## 5.1 CODE

```
a1=imread('ref_1.png');
b1=imread('cap_1.png');
c1=imread('ref_2.png');
d1=imread('cap_2.png');
e1=imread('ref_3.png');
f1=imread('cap_3.png');
%g1=imread('ref_4.png');
%h1=imread('cap_4.png');
```

```
a2=rgb2gray(a1);
b2=rgb2gray(b1);
c2=rgb2gray(c1);
d2=rgb2gray(d1);
e2=rgb2gray(e1);
f2=rgb2gray(f1);
%g2=rgb2gray(g1);
%h2=rgb2gray(h1);
```

```
a3=imresize(a2,[700 700]);
b3=imresize(b2,[700 700]);
c3=imresize(c2,[700 700]);
d3=imresize(d2,[700 700]);
e3=imresize(e2,[700 700]);
f3=imresize(f2,[700 700]);
%g3=imresize(g2,[700 700]);
%h3=imresize(h2,[700 700]);
```

```
edge_a=edge(a3,'sobel');
edge_b=edge(b3,'sobel');
edge_c=edge(c3,'sobel');
edge_d=edge(d3,'sobel');
edge_e=edge(e3,'sobel');
edge_f=edge(f3,'sobel');
%edge_g=edge(g3,'sobel');
%edge_h=edge(h3,'sobel');
```

```
figure,imshow(edge_a);
figure,imshow(edge_b);
figure,imshow(edge_c);
figure,imshow(edge_d);
figure,imshow(edge_e);
figure,imshow(edge_f);
%figure,imshow(edge_g);
```

```
%figure,imshow(edge_h);
```

```
m=0;  
wh=0;  
bl=0;
```

```
for a=1:1:700  
    for b=1:1:700  
        if(edge_a(a,b)==1)  
            wh=wh+1;  
        else  
            bl=bl+1;  
        end  
    end  
end
```

```
for i=1:1:700  
    for j=1:1:700  
        if(edge_a(i,j)==1)&(edge_b(i,j)==1)  
            m=m+1;  
        else  
            ;  
        end  
    end  
end
```

```
m2=0;  
wh2=0;  
bl2=0;
```

```
for a=1:1:700  
    for b=1:1:700  
        if(edge_c(a,b)==1)  
            wh2=wh2+1;  
        else  
            bl2=bl2+1;  
        end  
    end  
end
```

```
for i=1:1:700  
    for j=1:1:700  
        if(edge_c(i,j)==1)&(edge_d(i,j)==1)  
            m2=m2+1;  
        else  
            ;  
        end  
    end  
end
```

```

        ;
    end
end
end

```

```

m3=0;
wh3=0;
bl3=0;

```

```

for a=1:1:700
    for b=1:1:700
        if(edge_e(a,b)==1)
            wh3=wh3+1;
        else
            bl3=bl3+1;
        end
    end
end
end

```

```

for i=1:1:700
    for j=1:1:700
        if(edge_e(i,j)==1)&(edge_f(i,j)==1)
            m3=m3+1;
        else
            ;
        end
    end
end
end

```

```

% m4=0;
% wh4=0;
% bl4=0;
%
% for a=1:1:700
%     for b=1:1:700
%         if(edge_g(a,b)==1)
%             wh4=wh4+1;
%         else
%             bl4=bl4+1;
%         end
%     end
% end
% end
%
%

```

```

%
% for i=1:1:700
%     for j=1:1:700
%         if(edge_g(i,j)==1)&(edge_h(i,j)==1)
%             m4=m4+1;
%         else
%             ;
%         end
%     end
% end
% end

```

```

total=wh;
total_m=(m/total)*100;

```

```

total2=wh2;
total_m2=(m2/total2)*100;

```

```

total3=wh3;
total_m3=(m3/total3)*100;

```

```

% total4=wh4;
% total_m4=(m4/total4)*100;

```

```

if(total_m<=total_m2)&(total_m<=total_m3)
    if(total_m2<=total_m3)
        ans=[1,2,3]
    else
        ans=[1,3,2]
    end
elseif(total_m2<=total_m)&(total_m2<=total_m3)
    if(total_m<=total_m3)
        ans=[2,1,3]
    else
        ans=[2,3,1]
    end
elseif(total_m3<=total_m)&(total_m3<=total_m2)
    if(total_m<=total_m2)
        ans=[3,1,2]
    else
        ans=[3,2,1]
    end
else
    ans='Nothing'
end

```

(OR another code which detects percentage of car space in an image)

```
b = imread('sample3.jpg');
a = rgb2gray(b);
b1 = imread('sampl1.jpg');
a1 = rgb2gray(b1);
```

```
% insert function of OCR before evaluating further
```

```
f1 = fspecial('average',3);
figure,imshow('sample3.jpg'); %original image
title('Sample image')
```

```
med = medfilt2(a,[39 39]); %median filter on grayscale image
```

```
i = histeq(med) %hist equilization
```

```
i1 = i>25; %threshold
```

```
c = filter2(f1,i1,'same'); %average
```

```
med2 = medfilt2(i1,[19 19]); %median filter on thresholded image
```

```
% e1 = edge(a,'prewitt');
% e2 = edge(a,'canny');
% e3 = edge(a,'sobel');
% e4 = edge(a,'roberts');
% subplot(2,3,1), imshow(a)
% subplot(2,3,2), imshow(a1)
%figure,imshow(i)%histeqalisation image
%figure,imshow(i1) %threshold>100
%figure,imshow(c) %average filtered img
%figure,imshow(med) %median filter img
figure,imshow(med2) %median filter in i1 img
title('Enhanced image')
```

```
a=0
b=0
for i = 1:2080
    for j = 1:1564
        if med2(i,j)==0
            a=a+1
        else
            b=b+1
        end
    end
end
```

```
end  
end  
end
```

```
a=a*100/(2080*1564)  
b=b*100/(2080*1564)
```

```
%var a b c d for each image  
%checking black percentage in the image  
%prioritising variables and making a sequence
```

(The above given code requires almost 9 minutes to execute even with one of the latest processor)

## 5.2 Results and discussion

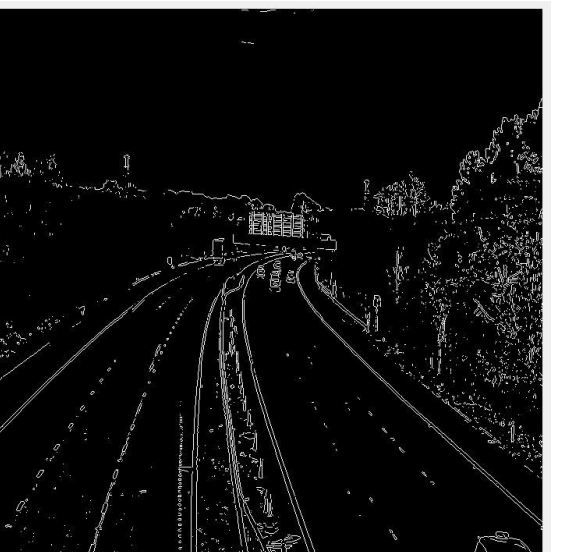
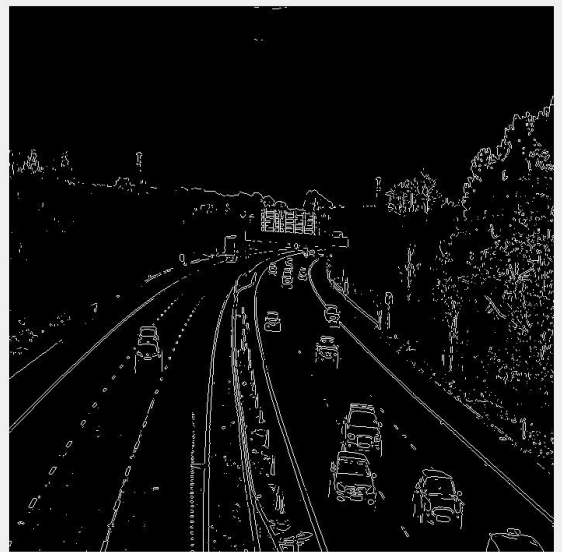
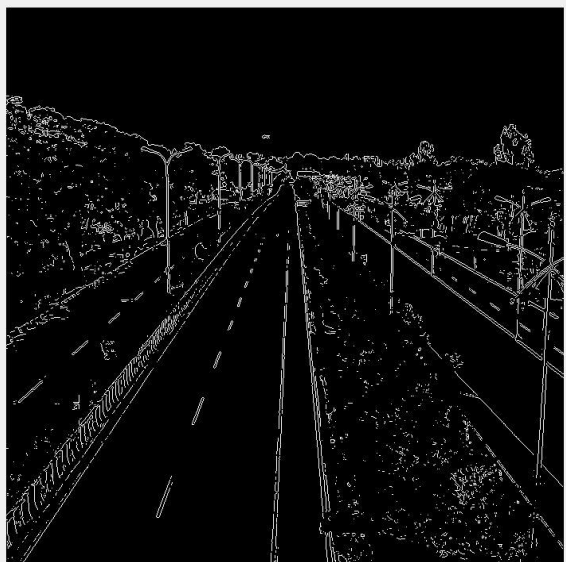
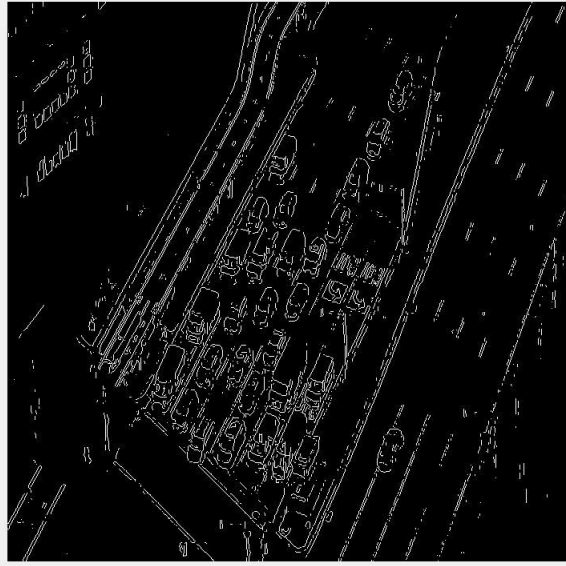
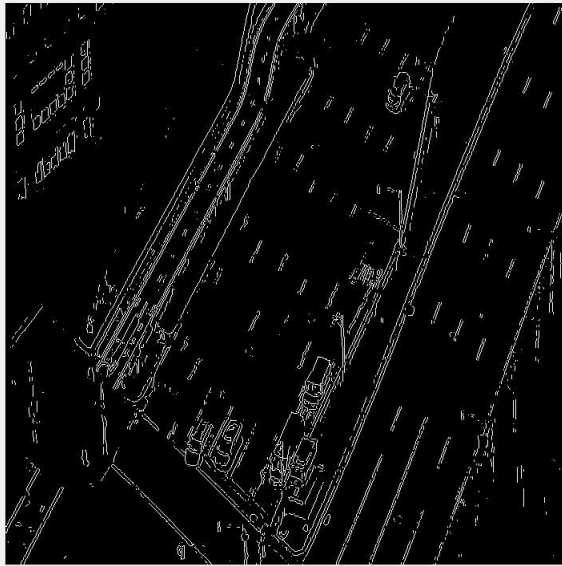
Due to unavailability of proper access to traffic cameras, some images are taken from the internet, and to create a reference image those images are photoshopped to remove cars present in the particular given image. And the compared with the original image so that one can have a basic idea of how the program works.



These are some images taken from the internet to have a rough idea of how a traffic camera captures an image.

These are the images which are photoshopped to look like the roads are empty for reference image, the example of captures images will be the original images.





These are the images received after edge detection, the images with empty roads are reference images, the images with vehicles are captured images.



total_m	87.8225
total_m2	92.7311
total_m3	74.8576

The above given is an image of variables, total\_m, total\_m2, total\_m3 which represents total matching percentage between images 5-6, 3-4 and 1-2 respectively.

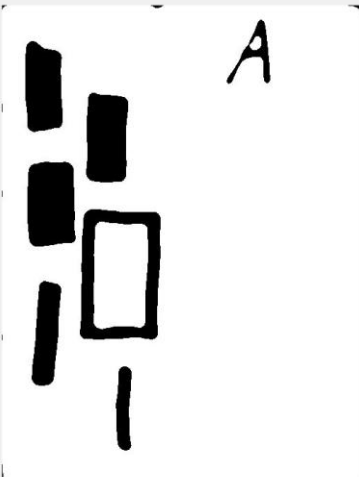
```
Command Window

ans =

     3     1     2

fx >>
```

The sequence is created, showing that traffic light of lane number 3 (images 1-2) should be green for highest amount of time, lane 2 should have minimum amount of green time, and lane 1 in mid-range length.

	<pre>Command Window  a =      10.9219  b =      89.0781  fx &gt;&gt;</pre>
--	--

The above given images represents an example of a road, with cars as black rectangles, bike as a thick line, and big box as a truck which matches the color of the road. 'a' is total percentage of black in the image, and 'b' is total percentage of white in the image.

## **6. Conclusion**

### **6.1 Conclusion**

“Traffic control using image processing” technique that we propose overcomes all the limitations of the earlier (in use) techniques used for controlling the traffic. Earlier in automatic traffic control use of timer had a drawback that the time is being wasted by green light on the empty. This technique avoids this problem. Upon comparison of various edge detection algorithms, it was inferred that Canny/Sobel Edge Detector technique is the most efficient one. The project demonstrates that image processing is a far more efficient method of traffic control as compared to traditional techniques. The use of our technique removes the need for extra hardware such as sound sensors. The increased response time for these vehicles is crucial for the prevention of loss of life. Major advantage is the variation in signal time which control appropriate traffic density using Image matching. The accuracy in calculation of time due to single moving camera depends on the registration position while facing road every time.

### **6.2 Future Work**

The focus shall be to implement the controller using DSP (Digital signal processing) as it can avoid heavy investment in industrial control computer while obtaining improved computational power and optimized system structure. The hardware implementation would enable the project to be used in real-time practical conditions. In addition, we propose a system to identify the vehicles as they pass by, giving preference to emergency vehicles and assisting in surveillance on a large scale.

## **7. References**

1. Digital image processing by Rafael C. Gonzalez and Richard E. Woods.

2. Ahmed S. Salama, Bahaa K. Saleh, Mohamad M. Eassa, "Intelligent Cross Road Traffic Management System (ICRTMS)," 2nd Int. Conf. on Computer Technology and Development, Cairo, Nov 2010, pp. 27-31.
3. B. Fazenda, H. Atmoko, F. Gu, L. Guan and A. Ball "Acoustic Based Safety Emergency Vehicle Detection for Intelligent Transport Systems" ICCAS-SICE, Fukuoka, Aug 2009, pp.4250-4255.
4. Z. Jinglei, L. Zhengguang, and T. Univ, "A vision-based road surveillance system using improved background subtraction and region growing approach," Eighth ACIS Int. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Qingdao, August 2007, pp. 819-822.
5. M. Siyal, and J. Ahmed, "A novel morphological edge detection and window based approach for real-time road data control and management," Fifth IEEE Int. Conf. on Information, Communications and Signal Processing, Bangkok, July 2005, pp. 324-328.
6. Y. Wu, F. Lian, and T. Chang, "Traffic monitoring and vehicle tracking using roadside camera," IEEE Int. Conf. on Robotics and Automation, Taipei, Oct 2006, pp. 4631– 4636.
7. Reulke, S. Bauer, T. D'oring, F. Meysel, "Traffic surveillance using multi-camera detection and multi-target tracking", Proceedings of Image and Vision Computing New Zealand 2007, pp. 175–180, Hamilton, New Zealand, December 2007.
8. [www.nzta.govt.nz](http://www.nzta.govt.nz)  
live traffic webcams / NZ Transport Agency
9. <http://www.mathworks.com> Web
10. Traffic light control system simulation through vehicle detection using image processing

By Mac Michael B. Reyes and Dr Eliezer A. Albaccea