Amrita Viswa Vidyapeetham Amrita School of Engineering, Bengaluru Department of Electronics and Communication Engineering

15ECE387-Open Lab

DENSITY BASED TRAFFIC LIGHT CONTROL USING IMAGE PROCESSING

Submitted by

REG NO.	**NAME**
BL.EN.U4ECE18121	PRIYANKA
BL.EN.U4ECE18507	K.M.DIKSHITHA
BL.EN.U4ECE18514	BRINDA HITENDRAPRASAD BRAHMBHATT

Faculty in charge

Dr. Ganapathi Hegde	**Dr. Jalpa Shah**
Assistant Professor(SG)	**Assistant Professor**

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 IMPORTANCE OF TRAFFIC REGULATION NEAR TRAFFIC SIGNAL

1.1.1 EXISTING TRAFFIC LIGHT RULES IN INDIA

The traffic light rules are universal and similar in every part of the world. The red-yellow-green traffic signal (3 lighted system) system means the same for people across the globe**.** A traffic signal comprises major parts like the lights, controller and sensor. Even though the traffic light rules in India or in other parts of the world are the same, the technologies used for the traffic signal setting might be widely different. The traffic lights used in India are basically **pre-timed** in the sense the time of each lane/street to have a green signal is fixed. In a four lane traffic signal one lane is given a green signal at a time. Thus, the traffic light allows the vehicles of all lanes to pass in a sequence. So, the traffic can advance in either straight direction or turn by 90 degrees.

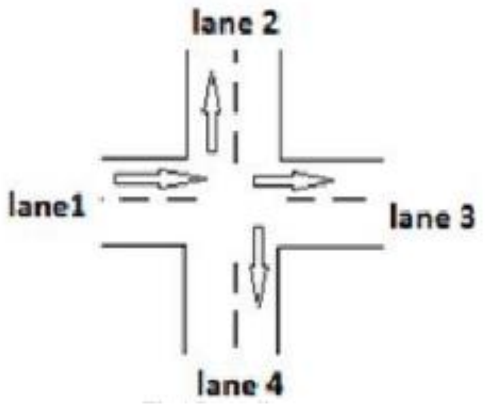


Figure1.1: 4-lane Road Schematic

1.1.2 PROBLEMS WITH THE EXISTING METHOD:

- Even if the traffic density in a particular lane is the least, one has to wait unnecessarily for a long time .
- When a traffic light shows a green signal it unnecessarily makes other lanes wait for even longer durations.

- Therefore in India, with the growing number of vehicles, traffic congestion at junctions has become a serious issue especially during peak hours.

SIGNAL COLOR	DESCRIPTION
RED	The vehicles must stop
YELLOW	The transition time between green and red color. It signifies that vehicles must slow down(if previously green) or must be ready to move(if previously red).
GREEN	The vehicle can go

1.1.3 PROPOSED SOLUTION

Vehicle detection plays an important role in designing intelligent traffic light systems. With a view to do improvements , it is proposed to develop an Automatic Vehicle recognition system using various image processing techniques. The system basically counts the number of vehicles in each lane /street and declares the densest road to get highest priority in order to reduce traffic congestion. This **simulated version** of this system uses the image(s) acquired from the camera installed near the traffic signal in order to realize the densest road. The foreground of the acquired image is detected using background subtraction technique . Later on morphological opening is applied to reduce the noise in the image. In the final step, the vehicles are detected using blob analysis. Same method is applied to all the four lanes . Priority is given to the lane with more number of vehicles. But, Using the images to realize the densest might not be a practical solution since it gives the result for that particular instant. So, to practically realize the scenario , video data is acquired from the stationary camera installed. This video is seen as a sequence of image frames . In order to detect the foreground of the video frame , Gaussian Mixture Model (GMM) is used . After detecting the foreground, objects(vehicles) are recognized using blob analysis. The results are encouraging as we got 95% of accuracy using these techniques.

1.2 OBJECTIVE

1.2.1 THINGS TO BE DONE USING IMAGE PROCESSING TOOLBOX

1. To read the acquired image using MATLAB and pre-process the image.

2. To create foreground mask of pre-processed image by background subtraction

3. To remove the noise using morphological opening(s)

4. To apply blob analysis to identify clusters

5. To analyze the output by comparing expected vs. detected vehicles

6. To conclude the densest road based on detection

1.2.2 THINGS TO BE DONE USING COMPUTER VISION TOOLBOX

1. To read the video using MATLAB and acquire video frames

2. To detect foreground using GMM

3. To do blob analysis

4.To analyse number of vehicles in each video frame

1.3 METHODOLOGY

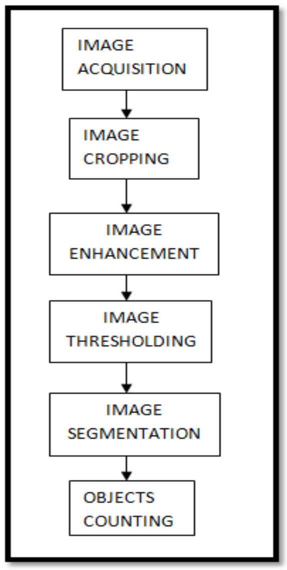


Figure 1.2: Design Flow

1.4 OUTCOMES

STREET NAME	**EXPECTED VEHICLES**	**DETECTED VEHICLES**	**ACCURACY(%)**
STREET 1	5	5	100
STREET 2	9	9	100
STREET 3	5	4	80
STREET 4	1	1	100

Average Accuracy = 95%

CHAPTER 2
DESCRIPTION OF TOOLS

2.1 LIST OF FUNCTIONS USED FROM IMAGE PROCESSING TOOLBOX

2.1.1 IMREAD( ) :

This function is used to read the filename that is specified within it's opening and closing parenthesis and assign it to a variable. This is used to read images. The syntax for it is as follows

variable_name = imread("file_name")

Here the variable assigned is nothing but a name given to that file that is used as a reference only within that program (or the scope of that variable). This function helps the program to understand what kind of file it is, as in the format of the file. It has many other arguments that help us analyze the image in a better way. Also, Depending upon the location of the file the file_name argument will change.The file_name should be enclosed within quotes.This function can retrieve files that are on the internet as well. We need to give the URL as the file name for that purpose. We can give the path of that file as the file name. We can also give the remote_location of the file as the file_name. The function will work for all them. If the file name attribute is an array or collection of images, the function picks the first image and assigns variable *variable_name* to it.

Eg : X = imread("example_image.jpg");

2.1.2 RGB2GRAY( ) :

This function is used to convert an RGB image to grayscale. It can also be used to convert a colour map from rgb to grayscale. The syntax for this is as follows

variable_name=rgb2gray("RGB_image")

The RGB image or the colour map is specified within opening and closing parenthesis as shown above. The variable used to refer to the grayscale image or colour map is written to the left of the assignment operator. This function removes the saturation of colours and the hues from the image but retains the brightness of the image. This is how the grayscale image is obtained. Same is done to get the grayscale equivalent of the input colour map.

Note : For colour map, the input is a matrix with N rows and 3 columns. N being the number of pixels and 3 columns corresponding to Red,Green and Blue respectively. Thus the syntax to read colour maps is different. However the syntax for rgb2gray remains the same. The output for the colour map is a grayscale colour map with 3 columns. The values in each column are identical though, and they represent the luminescence(or intensity) values for each pixel.

Eg : X = rgb2gray(“example_rgbimage.jpg”); {for rgb image}

[X1,colour_map] = imread(“cmap.tif”); {for colormap}

X2 = rgb2gray(X1);

2.1.3 STREL( ) :

This function forms a structuring element. It is used for morphological functions. This function basically creates a flat morphological structuring element which can be of N-dimensions, where N is 2 or more, which is then used for computation in operations like eroding, dilating, etc,. This function helps in making the image more uniform by removing noises like small holes, etc,. It creates a shape (can be rectangle, triangle, diamond, sphere, etc) that is the building block of above mentioned operations. This function has a lot many arguments in its syntax, the simple one being,

Variable_name = strel(neighbourhood)

We have used the below given syntax in our code,

Variable_name = strel(‘shape’,arguments)

Here the arguments describe the properties or attributes of the shape. This is a more detailed way of creating a morphological element. As we know the quality of input images and the quality of the object to be detected this type of definition will be advantageous to give accurate results.

Eg : X = strel(‘rectangle’,10,5);

2.1.4 IMRESIZE() :

This function is used to resize the image that is mentioned within its parenthesis. It modifies the dimensions of the image to a new dimension that is specified by us. Using this function it maintains a uniformity in the inputs given to the function over time. In our project, the input device is a camera, let’s say if the camera gets broken due to certain circumstances, and is replaced by a new one, this new camera will give images of different dimensions. Maintaining consistency in output irrespective of input conditions makes a program efficient and reliable. Hence this function increases the reliability of the code. The syntax for this function is as,

Variable_name = imresize(‘image’,[dimensions])

WE can use this function for a colour map as well. It uses antialiasing and interpolation concepts to resize the image whilst preserving the other properties of the image. This function has various other arguments depending upon the type of input, the interpolation method used, etc,. When no method is specified, it uses the default interpolation methods to resize the given image.

Eg : X = imresize('Image',10,5);

2.1.5 IMOPEN() :

This function is used to open an image, just like imread(), however, the difference here is that imread opens a file, whereas imopen opens a morphological image. This function is used to apply the above given strel function on an image and then display it. This function, in simple words, passes the image via a structuring element mentioned in its arguments and returns the output. It performs a morphological open operation which consists of two operations - erosion and dilation. These two operations use the same structuring element. Syntax for this function is

Variable_name = imopen('image',structuring element)

Another syntax for this is,

Variable_name = imopen('image',neighbourhood)

Here the neighbourhood describes the morphological structuring element neighbourhood. This can do two steps at a time, imopen() and strel(). For this the image can be RGB too. Whereas for the first syntax we usually perform binarization or convert to grayscale before applying morphological operations to it because the morphological operations work best on them.

Eg : X1 = imopen('Image',X);

X1 = imopen('Image',nbhd);

Where, Nbhd → neighbourhood

2.1.6 IMBINARIZE( ) :

This function converts grayscale image into a binary image by using the method of thresholding. It can be applied on 2 dimensions or 3 dimensions. A grayscale image forms the 2-D input whereas a volume forms 3-D input. Since in our project we are working with 2-D inputs (Images) we will have to turn our images into grayscale before binarising it. As it can be see that the functions are all interlinked, not in the sense of their working principles, but their functionalities. That is, the output of one function becomes input to another, to use one function we must bring the image to such a stage that it is acceptable as the input to that function, which again requires the other functions. The simplest syntax for this function is, *Variable_name = imbinarize(image)*

Here, *image* specifies the grayscale image as input. There can be many other arguments that can be included in this function that specifies the method for thresholding, threshold value or the method of adaptive thresholding. These arguments let us describe the binarization process in much detail to suit our purpose.

Eg : X1 = imbinarize(X);

2.1.7 BWAREAOPEN( ) :

This is a very simple function that is used to remove small objects from the image that are unwanted. Basically it is used to remove small noises from the image so that we can focus on detecting our main object easily. The syntax for this function is

Variable_name = bwareaopen(image,threshold_value)

Here we specify the threshold value of pixels in the arguments. If the connected group of pixels in the image are smaller in number than threshold value, it is removed, only the bigger groups are retained and analysis or object detection is done in that.

Eg : X1 = bwareaopen(X,70);

2.1.8 IMDILATE( ) :

This function dilates the image. It doesn't perform any other operation. This function is solely used to expand or dilate the image's pixels by using a structuring element, and to give specific values for the operation of dilation. The syntax is as follows

Variable_name = imdilate(image, structuring_element)

The structuring element here is the output of the strel function. It can also be an output of offsetstrel function but it is unnecessary in our code and hence is not used. This function, imdilate(), adds pixels to boundaries of the object. It removes sudden changes in intensities of the image by increasing the neighbouring values. This is done so that there is more uniformity in our image and object detection becomes easier.

Eg : X1 = imdilate(image, X);

Where, X → structuring element

2.1.9 IMERODE( ) :

This function is used to apply the process of erosion on an image. This function contracts the image pixels. It can be said that it performs reverse operation of dilation, although not in exact terms. The process of erosion is applied on the image with respect to the structuring element as mentioned in the arguments of the calling function statement. Using this, the morphological operation is carried out to isolate the object we need to detect from the background. The syntax for this function is similar to imdilate() function

Variable_name = imerode(image, structuring_element)

This function has other arguments too, to make the output image more and more accurate. The structuring element referred in the syntax is the output of strel() function. The other syntaxes may or may not include the strel function. Same goes with imdilate() function. However, according to our implementation, strel() function gives the most accurate results and hence is used for implementation.

Eg : X1 = imerode(image, X);

Where, X→ structuring element

2.2 LIST OF FUNCTIONS USED FROM COMPUTER VISION TOOLBOX

2.2.1 VISION.FOREGROUNDDETECTOR( ) :

This function is used to detect and separate the foreground of the image from the background. It is useful in detecting the object of our choice as it limits the area of search and makes it easier to locate. It creates a mask or a subset of the image by applying Gaussian Mixture Model. The simplest syntax of this function is

Variable_name = vision.ForegroundDetector

This line will just create a mask of the default values using Gaussian Mixture Models. When this function is called, the mask that it creates will lock its properties after creation. Hence once it is created we cannot undo it by calling another function. Therefore, to create a mask of our choice we have many arguments that can be used for this function that specify the type of the mask we want to create.

Variable_name = vision.ForegroundDetector('Name_of_property', value)

This is helpful as we can monitor or change many values of the foreground based on our application. This will help in further reducing the search area and optimizing the code.

Eg :   X1 = vision.ForegroundDetector(‘NumTrainingFrames’,10);

2.2.2 VIDEOREADER( ) :

Just like imopen() function is used to read image files, VideoReader() function is used to video files. This function basically creates an object first, then it reads the data of the video file into the object, which is then accessible by you within the function using that object. The syntax for this function is

Variable_name = VideoReader(‘File_name’)

There are other arguments to this function that specify what part of the video should be taken or height of the video frame, etc,. Since we need the entire for analysis we do not give any arguments. We can also call other functions on this variable_name to get the properties of the video frame, for verification purposes.

Eg :   X1 = VideoReader(‘lane1.mp4’);

2.2.3 VISION.BLOBANALYSIS( ) :

This function is used to apply the operations or method of blob analysis. This method will be discussed in later sections. For now, it is important to understand that this function tracks all the connected parts of an image and sets them as a bundle, referring to it as one single entity. It is applied only to binary images, since only then can detection of blobs be possible. The syntax for this is similar to that os vision.ForegroundDetector() function,

Variable_name = vision.BlobAnalysis

This function simply returns a blob analysis object that can be used to compute connected parts within an image. Another syntax for this is,

Variable_name = vision.BlobAnalysis(‘Name_of_property’, value)

Similar to vision.ForegroundDetector(), here too we can have properties as arguments to specify the object more clearly, to suit our application. The sole purpose of this is to get accurate results and increase efficiency.

Eg :   X1 = vision.BlobAnalysis(‘Connectivity’,4);

2.2.4 INSERTTEXT( ) :

This function is used to insert text in an image or video. It can be used to display the output next to the output video or image instead of displaying it on the matlab window. The syntax for this is

Variable_name = insertText(Image, position, Text_to_be_displayed)

Here, position specifies where in the image do we want the text to appear. There are many other arguments along with this, and many other syntaxes too. We can insert numeric value and textual value in the image or video. We can give different colours to the text and text boxes to differentiate between the image and the text.

Eg : X1 = insertText(X, [1,1], numCars);

Where, X → image

[1,1] → top left corner

numCars → The variable that contains the value to be displayed

CHAPTER 3
DESIGN AND DEVELOPMENT

3.1 IMAGE PRE-PROCESSING

The acquired image from the CCTV is read into MATLAB workspace. The image is resized to the required amount because we want to analyze all the 4 images, each representing one street/lane to be of same size. We resize in such a way that , the image gets expanded rather than compressing because compressing might lead to wrong detection of the image .For further analysis, we first convert the RGB image to Gray Scale image using **rgb2gray** function.

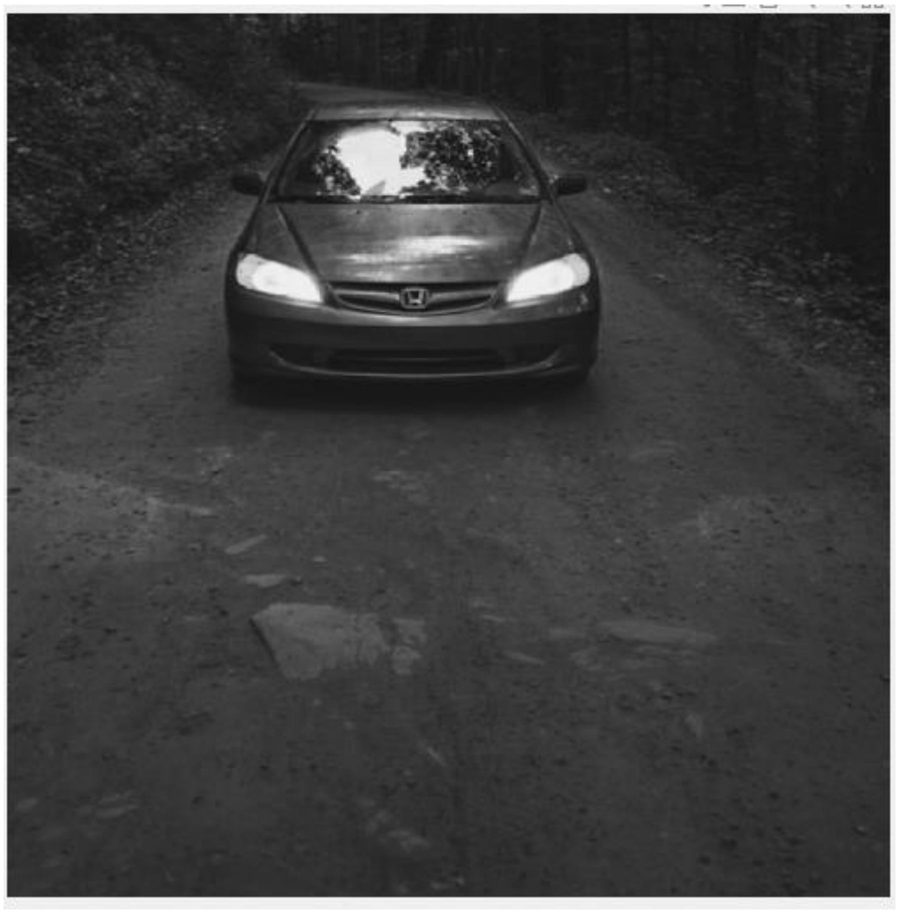


Figure3.1: Gray-Scale Image

3.2 FOREGROUND DETECTION AND THRESHOLDING

First, we are designing a structuring element using **strel** function. Here we are generating a disk of radius 50 pixels which is enough to span the foreground of the image. Next, we are extracting the background of the image using **imopen** function which basically opens the image by first dilating and then eroding using the same structuring element (disk) that we defined**.** This extracted background approximation image is subtracted from the original image to get the foreground of the image. But the resulting image is bit dark for analysis . Therefore we use imadjust to increase the contrast of the processed image by saturating 1% of the data at both low and high intensities and by stretching the intensity values to fill the uint8 dynamic range. To analyze the image further using image tool box, we need to have logical pixel values .So we binarize the image using **imbinarize** function .We remove the noise present in this binarized image using morphological operations. (Here we eliminated all the pixel values lesser than 50 using **bwareaopen** function).This is part of image thresholding.

Figure3.2: Background Image

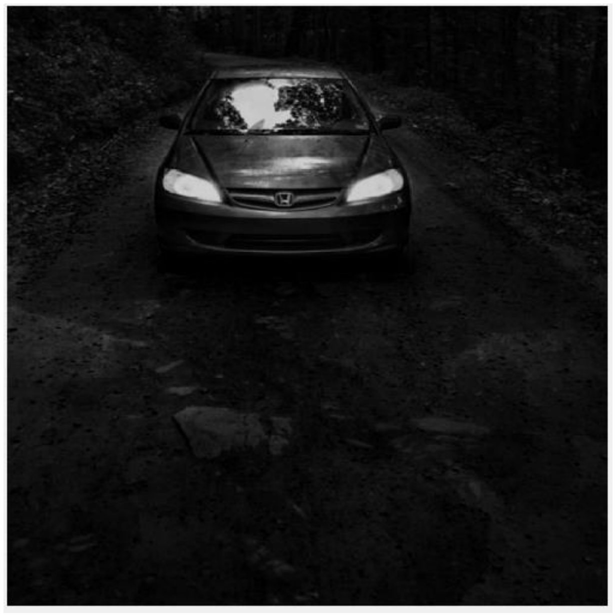


Figure3.3: Foreground Image

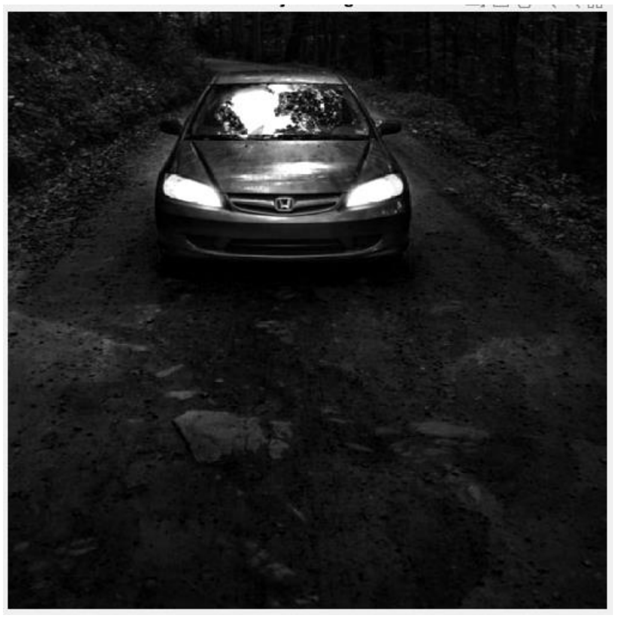


Figure3.4: Adjusted Foreground Image

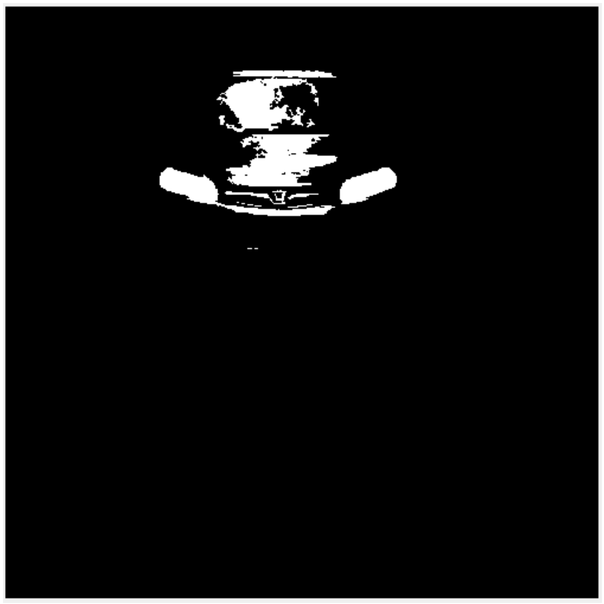


Figure3.5: Binary Image

3.3 FILTERED FOREGROUND

In real time scenario, there are things other than vehicle(s) that is highlighted in the image. We have to make sure that the unwanted things (White lane marking, broken white lines, and Solid yellow line) are removed and only the desired vehicle(s) is highlighted for counting. To achieve this, basic morphological operations of dilation and erosion are performed using different structuring element. For eroding rectangle of 4*15 pixels is used and for dilation a square of 50 pixels is used.

3.3.1 DILATION

Dilation adds the pixel values to the boundaries of the object to the image. The amount of dilation depends on the structuring element that we define. The value of the output pixel is the *maximum* value of all pixels in the neighborhood. In a binary image, a pixel is set to 1 if any of the neighboring pixels have the value 1.

Morphological dilation makes objects more visible and fills in small holes in objects. For understanding purpose, look at the following image.

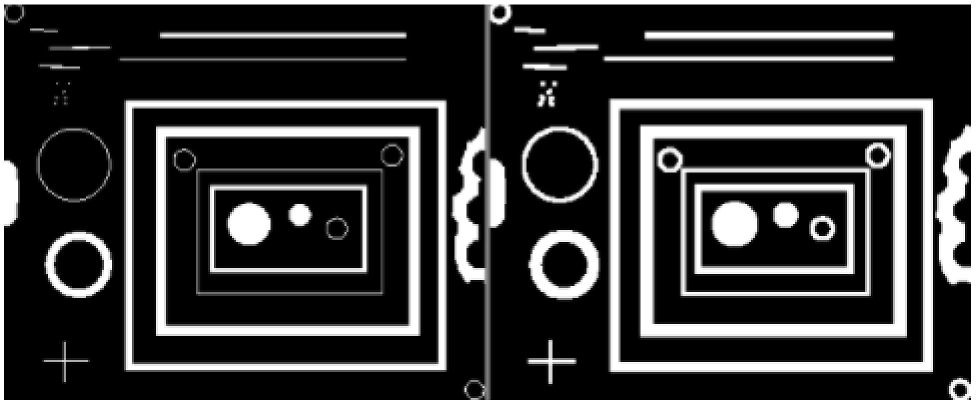


Figure3.6: Comparison of the original (left) and dilated image (right)

The dilation function applies the appropriate rule to the pixels in the neighborhood and assigns a value to the corresponding pixel in the output image. In the figure, the morphological dilation function sets the value of the output pixel to 1 because one of the elements in the neighborhood defined by the structuring element is on.

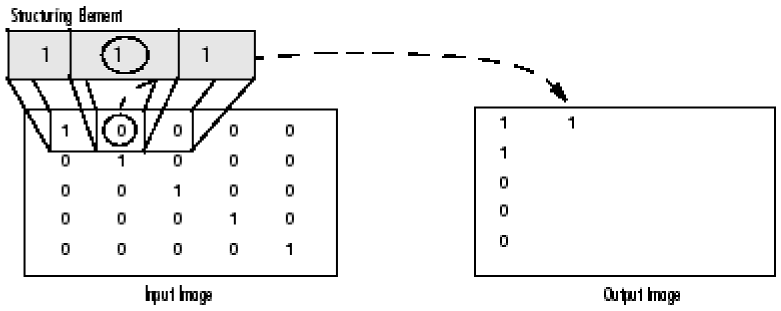


Figure3.7: Morphological Dilation of a Binary Image

3.3.2 EROSION

Erosion removes the pixel values to the boundaries of the object to the image. The value of the output pixel is the *minimum* value of all pixels in the neighborhood. In a binary image, a pixel is set to 0 if any of the neighboring pixels have the value 0.

Morphological erosion removes islands and small objects so that only substantive objects remain. Consider below image for understanding purpose.

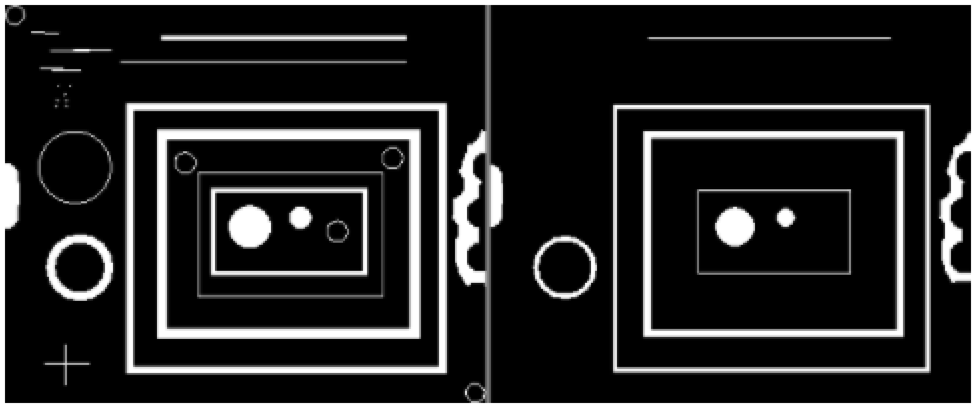


Figure3.8: Comparison of the original (left) and eroded image (right)

So, after applying these dilation and erosion techniques on the foreground image , the small objects in the image like white lines on the road or trees or dividers near the road are eroded and only the vehicle(s) are dilated. This makes the counting process of vehicle easy in which we apply blob analysis.

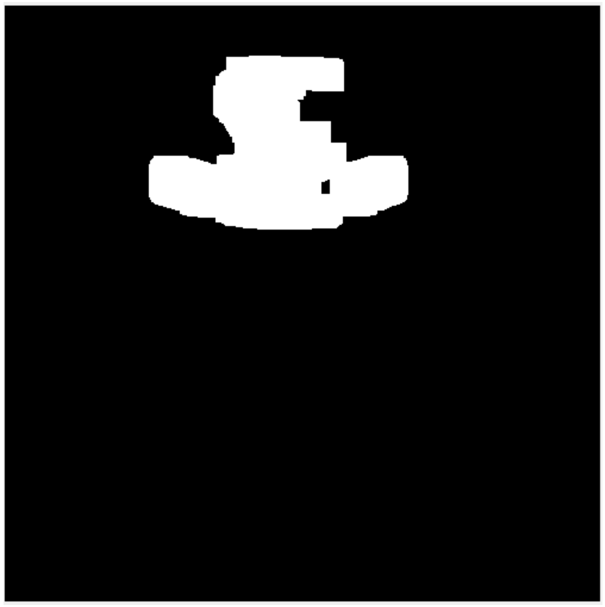


Figure3.9: Clean foreground image

3.4 BLOB ANALYSIS

The process of analyzing the binarized image is called blob analysis. Blob refers to group of pixel values (a lump) .This is the most basic method to find features of an object like shape, presence, number, area, position. The vision.blobanalysis function is used to find the bounding boxes of each connected component corresponding to the vehicle(s). This further filters the image by rejecting the pixel values lesser than 500 pixel. The number of bounding boxes corresponds to the number of vehicles in the image. The no. of cars found is denoted at the top left corner of the image.

For practical scenario, where we use a video of say 10 seconds to realize the vehicle(s) in particular lane/street, GMM method following with blob analysis gives efficient result.

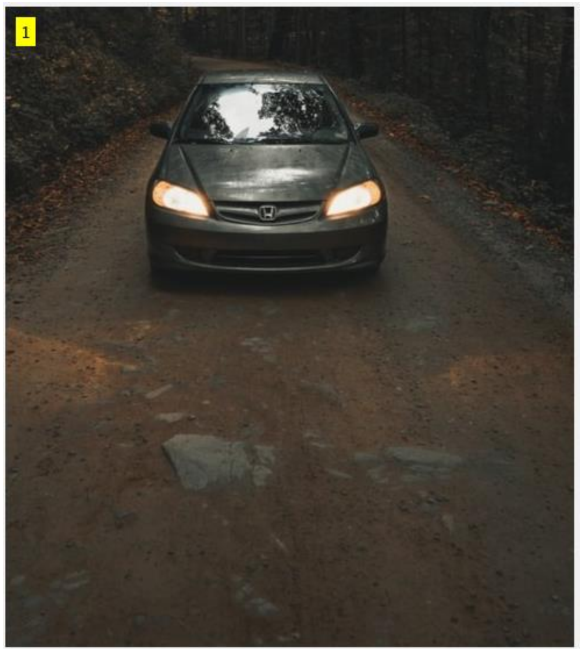


Figure3.10: Final image showing the number of detections

3.5 GAUSSIAN MIXTURE MODELS

GMM is a clustering method. Clusters are the set of data points which share common characteristics. Clustering method finds the se of data points that are closer. For every cluster there are predefined values like mean, variance which are basically used to group the clusters. Now there are many methods to do this clustering process like

- K-means clustering
- Hierarchical clustering
- GMM.

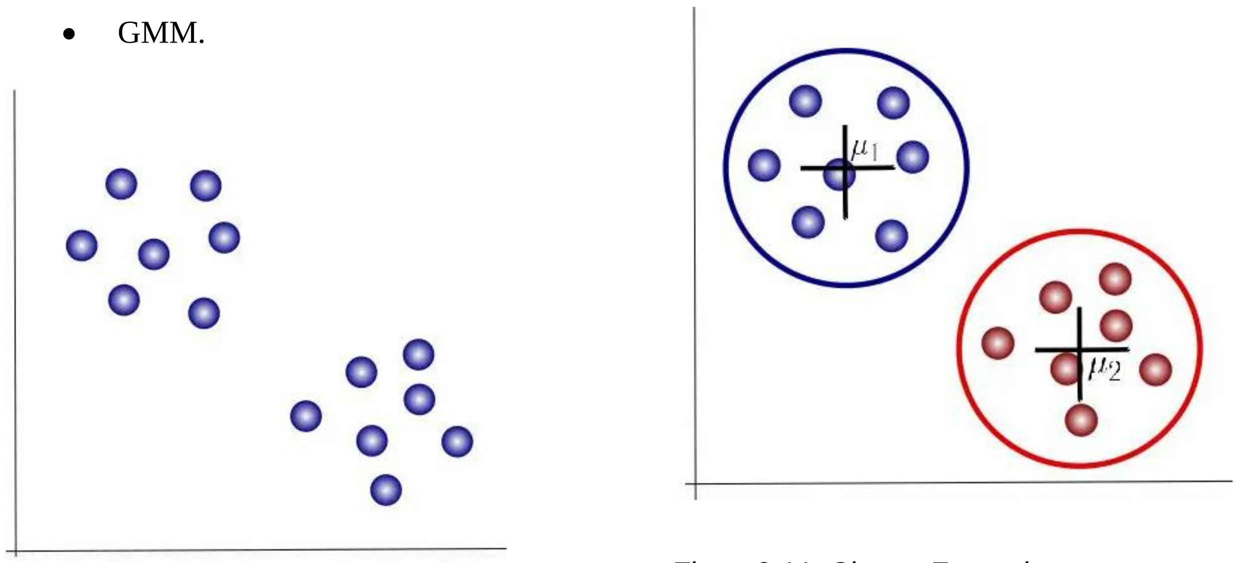


Figure3.11: Cluster Formation

K-means is a hard clustering method which means that it associates each point to one and only cluster. The limitation of this method is that there is no probability that tells us how much a data point is associated with the cluster whereas in GMM we get to know in how many clusters the data points are included so that we can subtract the repeated data points and reduce discrepancy . Gaussian Mixture is a function that includes multiple Gaussians equal to the total number of clusters formed. Each Gaussian in the mixture carries some parameters which are-

1. A mean that defines the center.
2. A covariance that defines the width.
3. A probability.

Therefore we opt for GMM model. The foreground detector uses GMM to detect clusters. Here we are using 3 GMM models and initializing the training frames to be 50.

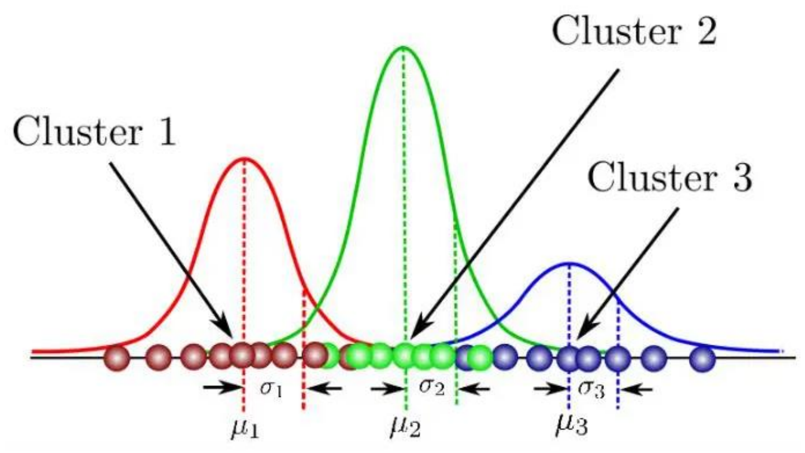


Figure3.12: Cluster Formation in GMM Model

CHAPTER 4
DEMONSTRATION AND RESULT

4.1 MATLAB CODE FOR ANALYSING 4 LANES

```
clc;
clear;
close;

%Handle to store all the files of jpg format in the given folder
files = dir('*.jpg');
%Array to store no. of vehicles in each street
a = [];
%variable to store the street number
pos = 1;
%Array to store sorted array of 'a'
b = [];
%iteration variable
j=1;
for i = 1:numel(files)

i1=imread(files(i).name); %Reads the image
i=imresize(i1,[1000,1000]);%Resize the image to 1000*1000 pixels
%expanding is preferred over compressing
i=rgb2gray(i); %convert RGB image to GrayScale image    %most
functions work on grayscale image

%figure; imshow(i); title('Street');

se=strel('disk',50);  % Defining the structuring element
background = imopen(i,se);  %opens the image morphologically

%figure;imshow(background);title('Background');

I2 = i - background;  %Subtracting background inorder to get
foreground
%figure;imshow(I2);title('fg');
I3 = imadjust(I2);%saturates the image   %Adjust the image to
brighten the foreground
%figure;imshow(I3);title('adjusted fg');
```

```
 i=imbinarize(I3);  %Binarize the adjusted image
i = bwareaopen(i,50);%to remove noise   %Morphologically open
the image to remove small holes in binary image
 %figure; imshow(i); title('binarised');

se = strel('rectangle',[4,15]);%4,15  %Defining the structuring
element
filteredForeground= imerode(i,se);  %Removing small objects
se1 = strel('rectangle',[50,50]);%50,50  %Defining the
structuring element
filteredForeground= imdilate(filteredForeground,se1);  %fills
the small holes
%%figure;imshow(filteredForeground);title('clean fg');


blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort',
true, ...
    'AreaOutputPort', false, 'CentroidOutputPort', false, ...
    'MinimumBlobArea', 500);%100 for dense areas 150 is original
value 80 for street-3
     %Defining blob of min area 500 pixels
bbox = step(blobAnalysis, filteredForeground);  % Generating
Boundingbox across detected blob


%Display number of vehicles found in image
numCars = size(bbox, 1)
result = insertText(i1,[10 10], numCars, 'BoxOpacity', 1, ...
    'FontSize', 14);
figure; imshow(result); title('Detected Cars');


%Store vehicles from each street in array
a(j) = numCars;
j = j+1;


end

%to find the densest street
densest = a(1);
```

```
for k = 2:4
   if(a(k)>a(k-1))
       densest = a(k);
       pos = k;
   end
end
%Printing the array of number of vehicles and the max number of
vehicles
a
densest

%printing the output
fprintf("\n Street %i = green \n ", pos);

%sorting the array in priority order
b=sort(a,'descend');
b

% fprintf("\n Street %i = green \n ", b(1));
% fprintf("\n Street %i = yellow \n ", b(2));
% fprintf("\n Street %i = red \n ", b(3));
% fprintf("\n Street %i = red \n ", b(4));


%printing the priority order
for k = 1:4
   for j = 1:4
       if(a(j)==b(k))
         fprintf("\n Street %i has %i cars ",j,b(k));
       end
   end
   if(k==1)
      fprintf("\n Light --> Green \n ");

   elseif(k==2)
      fprintf("\n Light --> Yellow \n ");

   else
      fprintf("\n Light --> Red \n ");
   end
end
```

RESULT:

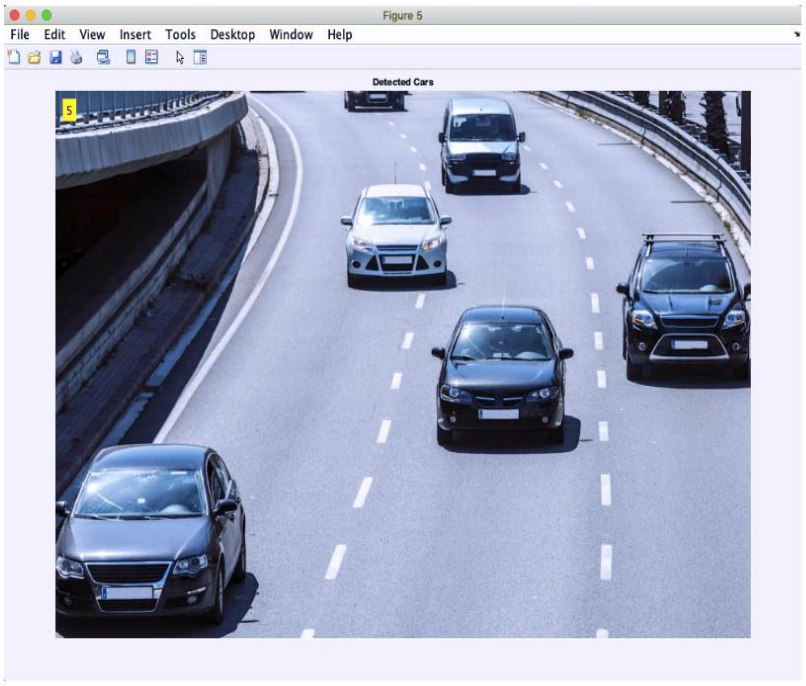


Figure4.1: Street 1

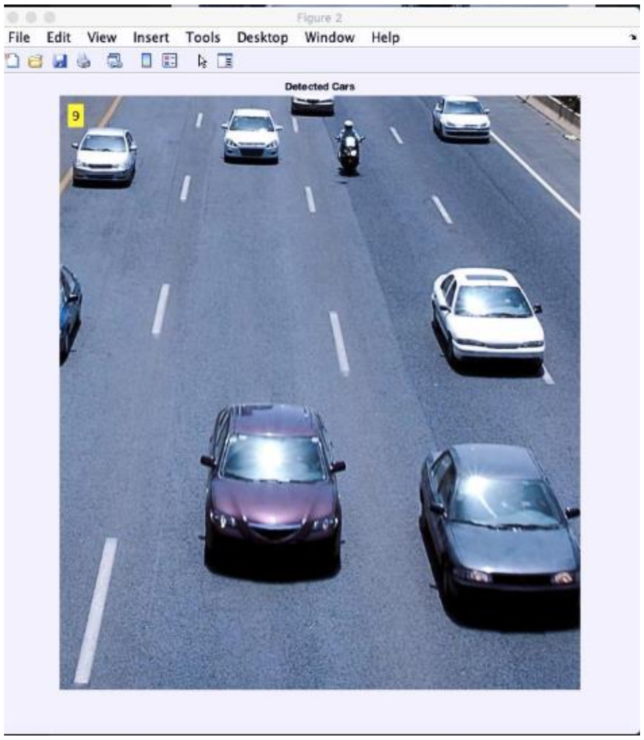


Figure4.2: Street 2

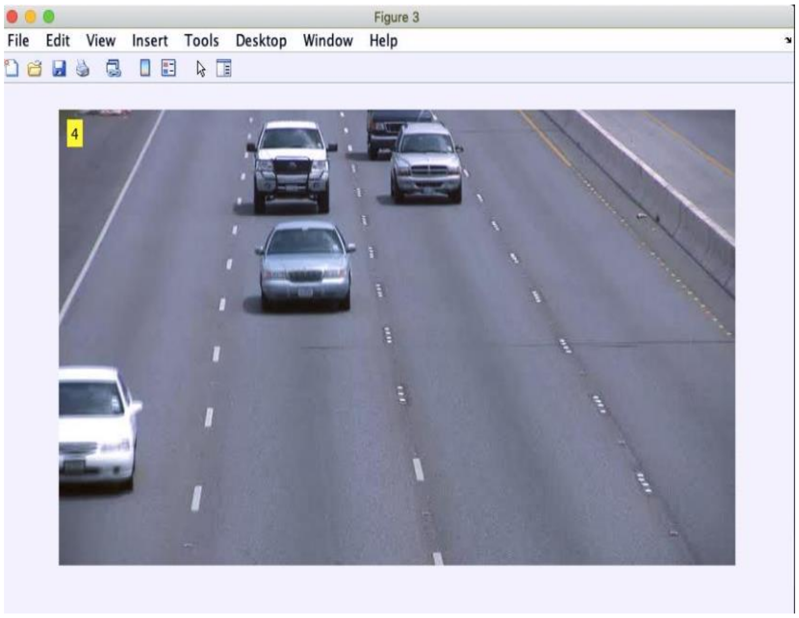


Figure4.3: Street 3

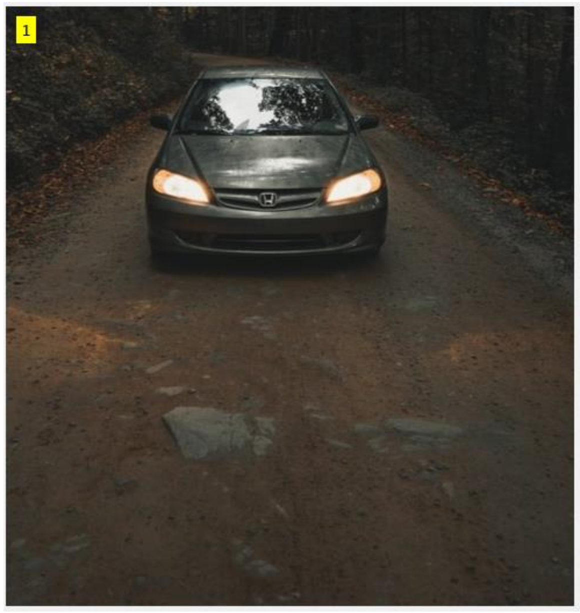
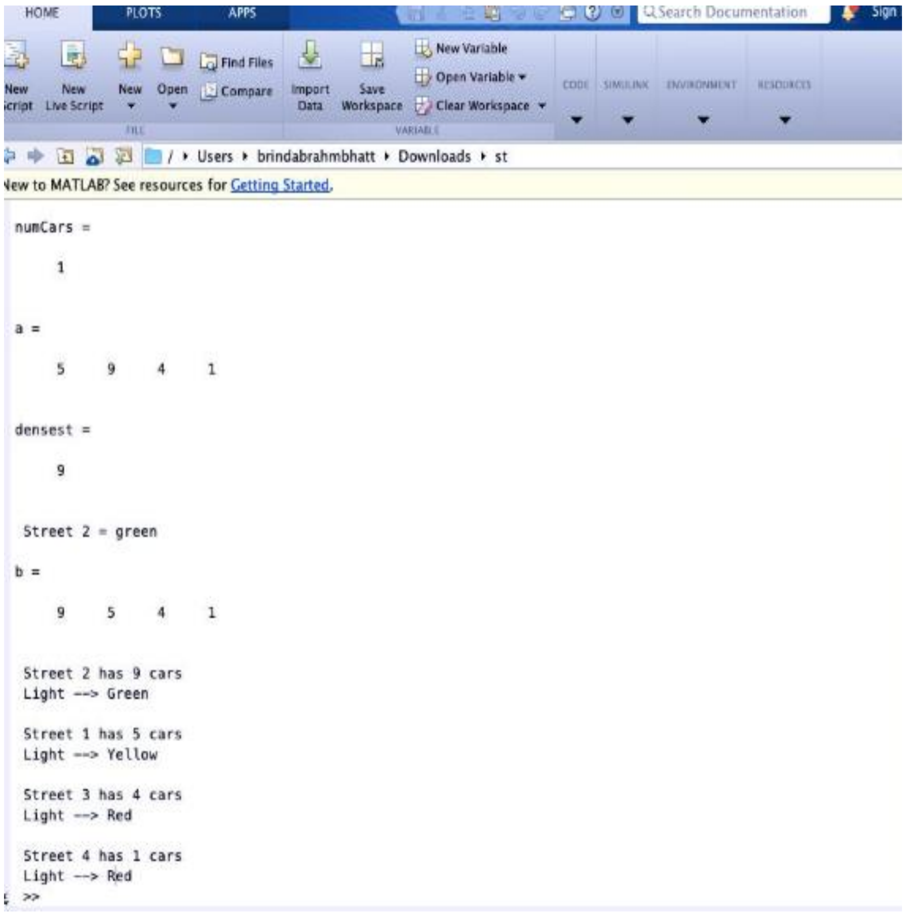


Figure4.4: Street 4



```
numCars =

     1


a =

     5     9     4     1


densest =

     9

 Street 2 = green

b =

     9     5     4     1

 Street 2 has 9 cars
 Light --> Green

 Street 1 has 5 cars
 Light --> Yellow

 Street 3 has 4 cars
 Light --> Red

 Street 4 has 1 cars
 Light --> Red
 >>
```

Figure4.5: Output

4.2 MATLAB CODE FOR ANALYSING VIDEO

```
%Initialising the foreground detector to 3 Gaussian models and
training
%frames of 50(first 50 frames)
foregroundDetector = vision.ForegroundDetector('NumGaussians',
3, ...
    'NumTrainingFrames', 50);

%Read the videoframe to the workspace
videoReader = VideoReader('street2.mp4');

%Loop to read all the frames in given video
```

```
for i = 1:150
    frame1 = readFrame(videoReader); % read the next video frame
    frame=imresize(frame1,[850,850]);%Resizing all the video
frames to same size
    foreground = step(foregroundDetector, frame);
end

%Computing the number of vehicles in first video frame
figure; imshow(frame); title('Video Frame');
figure; imshow(foreground); title('Foreground');

se = strel('rectangle', [50,50]);%Defining structuring element
filteredForeground = imdilate(foreground, se);%fill in the small
holes
se1=strel('rectangle',[80,80]);%Defining structuring element
filteredForeground = imerode(filteredForeground, se1);%removing
small objects
figure; imshow(filteredForeground); title('Clean Foreground');

blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort',
true, ...
    'AreaOutputPort', false, 'CentroidOutputPort', false, ...
    'MinimumBlobArea', 500);%Defining blob of minimum area of
500 pixels
bbox = step(blobAnalysis, filteredForeground);


%Displaying number of cars in each videoframe
numCars = size(bbox, 1);
result = insertText(frame, [10 10], numCars, 'BoxOpacity', 1,
...
    'FontSize', 14);
figure; imshow(result); title('Detected Cars');

%Computing vehicles in rest of the video frames
videoPlayer = vision.VideoPlayer('Name', 'Detected Cars');
videoPlayer.Position(3:4) = [650,400];  % window size: [width,
height]
se = strel('rectangle', [50,50]); % morphological filter for
noise removal

while hasFrame(videoReader)

    frame1 = readFrame(videoReader); % read the next video frame
    frame = imresize(frame1,[850,850]);
    % Detect the foreground in the current video frame
    foreground = step(foregroundDetector, frame);
    % Use morphological opening to remove noise in the
foreground
    filteredForeground = imdilate(foreground, se);
    se1=strel('rectangle',[80,80]);
   filteredForeground = imerode(filteredForeground, se1);

    % Detect the connected components with the specified minimum
area, and
    % compute their bounding boxes
    bbox = step(blobAnalysis, filteredForeground);

    % Draw bounding boxes around the detected cars
    result = insertShape(frame, 'Rectangle', bbox, 'Color',
'green');

    % Display the number of cars found in the video frame
    numCars = size(bbox, 1);
    result = insertText(result, [10 10], numCars, 'BoxOpacity',
1, ...
        'FontSize', 14);

    step(videoPlayer, result);  % display the results
end
```

RESULT (VIDEO IS INCLUDED IN SLIDES)

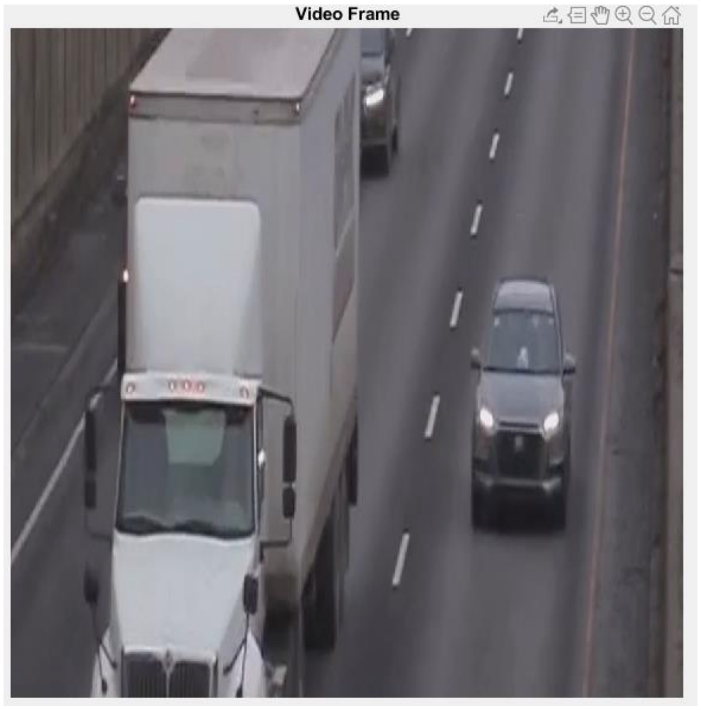


Figure4.5: Video frame

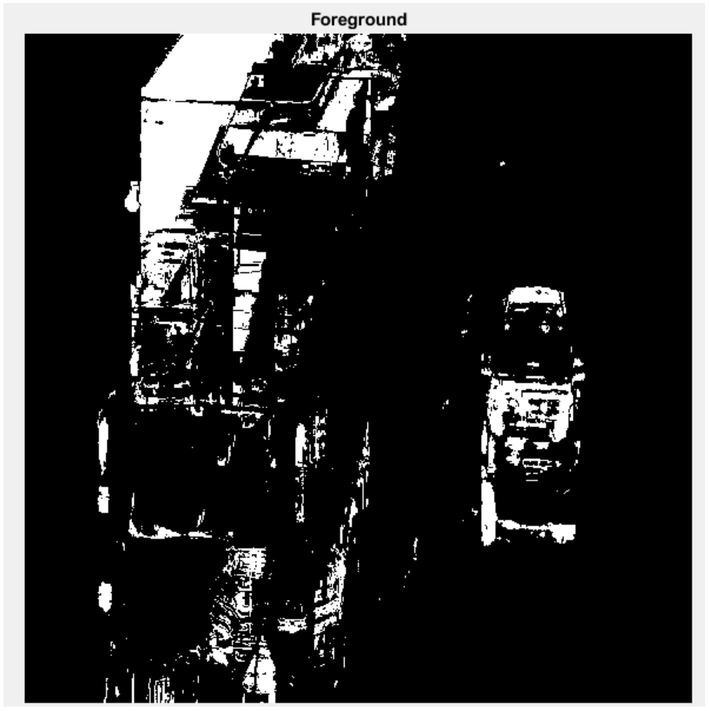


Figure4.6: Foreground frame

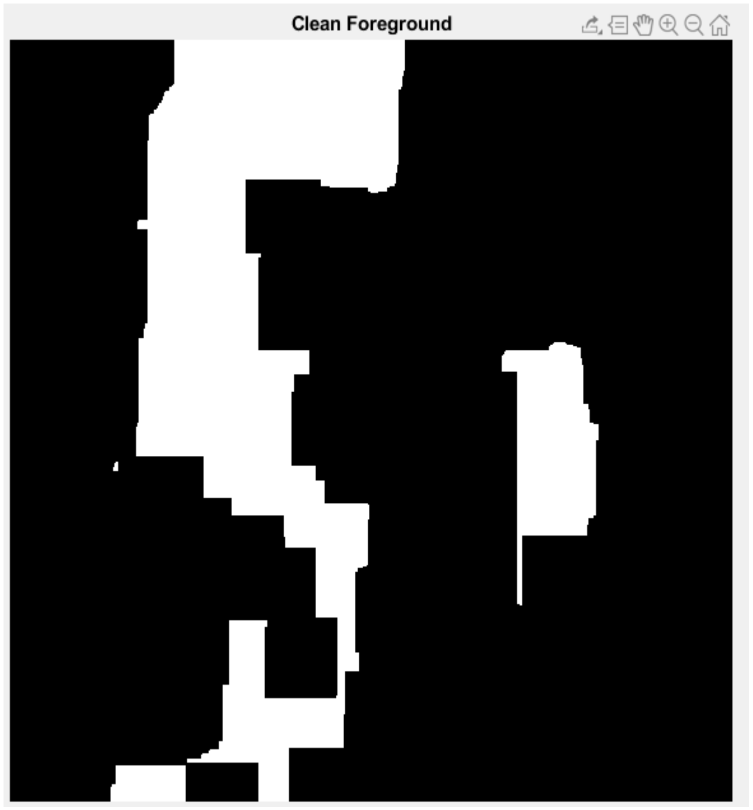


Figure4.7: Clean Foreground frame

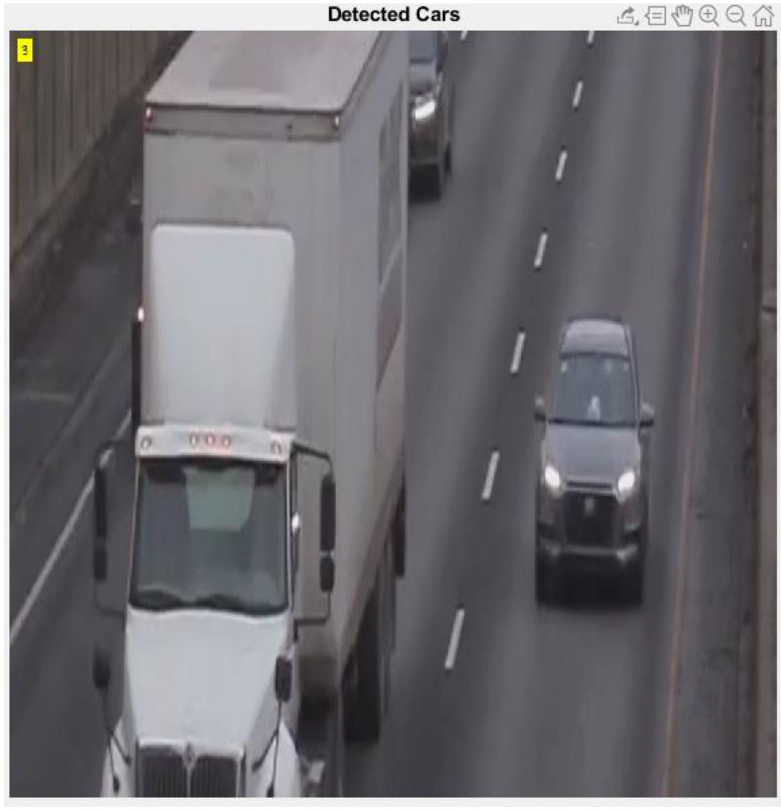


Figure4.8: Number of detections in the chosen frame

CHAPTER 5
CONCLUSION

The objective of this project is to detect number of vehicles in each lane and analyze the densest road so that traffic congestion near traffic signal can be reduced. In order to justify the mentioned statement this project is analyzed for both images(s) as well as real time environment (video). The image based analysis is done using background subtraction technique followed by morphological operation technique. The number of vehicles in the image is detected using blob analysis. Finally, based on the number of vehicles, the densest road is concluded and accordingly green-yellow-red signals are allocated to that. The video based analysis is carried through foreground detection based on GMM model in which we are using 1st 50 frames as training set for GMM. Also for detection purpose, the same BLOB technique used in the image analysis is utilized. With this technique, the number of vehicles in each frame for the given video sequence is detected. The detection accuracy for the proposed model is 95%.

As the proposed model is in its simplest yet accurate form, the future advancements to the project can be done by utilizing various other technologies like AI, Machine Learning, Big Data, CNN, YOLO techniques etc., So many other enhancements can be added to the model so as to utilize this project to its full. When this project is implemented in real time, a lot of time caused from the traffic congestions can be saved. This project is cost efficient and does not require a lot complex infrastructure. When Indian traffic conditions are considered, this project would be very efficient. Also as part of future enhancements, solar or wind energy source can be used in such a way that energy conservation will be utilized to its maximum and no energy would be wasted.

REFERENCES

[1]] Rahishet, Aparajita Sahoo, Aparna Indore, Vaibhav Deshmukh, Pushpa U S,” Intelligent traffic light control using Image processing ”

[2] Song Yuheng, Yan Hao, “Image Segmentation Algorithms Overview”

[3] Muhammad Waseem Khan ,”A Survey: Image Segmentation Techniques”, International Journal of Future Computer and Communication,Vol.3,No.2, April 2014

[4] Senthilkumaran N, Rajesh R. Edge detection techniques for image segmentation–a survey of soft computing approaches[J]. International journal of recent trends in engineering, 2009, 1(2): 250-254.

[5] Pal N R, Pal S K. A review on image segmentation techniques[J]. Pattern recognition, 1993, 26(9): 1277- 1294

[6] Haralick R M, Shapiro L G. Image segmentation techniques[J]. Computer vision, graphics, and image processing, 1985, 29(1): 100-132.

[7] K Adi , A P Widodo , C E Widodo , A Pamungkas , A B Putranto,” Automatic vehicle counting using background subtraction method on gray scale images and morphology operation”, IOP Conf. Series: Journal of Physics: Conf. Series 1025 (2018) 012025

[8]https://www.researchgate.net/publication/271424684_Image_processing_based_vehicle_detection_and_tracking_method

[9] https://in.mathworks.com/help/images/morphological-dilation-and-erosion.html

[10] https://www.cronj.com/blog/gaussian-mixture-model-in-image-processing-explained/