Application of Image Processing in Traffic Control System & License Plate Detection

Thesis report submitted to the Department of Electrical & Electronic Engineering of

BRAC University

By

Aminul Islam - 11321026

Nafiz Ahmed - 12221002

Toaha Hasnain - 12221052

Iffat Tabassum - 12121142

Supervised by

Avijit Das

Lecturer

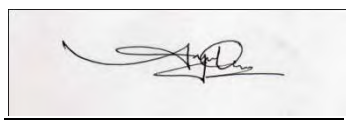
Department of Electrical and Electronic Engineering BRAC University, Dhaka

In partial fulfillment of the requirements for the Bachelor of Science degree in Electrical and Electronics Engineering Summer 2017 BRAC University, Dhaka

DECLARATION

We hereby declare that the thesis title "Application of Image Processing in Traffic Control System & License Plate Detection" submitted to the Department of Electrical and Electronics Engineering is our own work and has not been presented elsewhere for assessment. Where material has been used from other sources it has been properly acknowledged referred.

Signature of Supervisor



Avijit Das

Signature of Authors

_____

Aminul Islam

_____

Nafiz Ahmed

_____

Toaha Hasnain

_____

Iffat Tabassum

ACKNOWLEDGMENT

We would like express our heartiest gratitude towards our supervisor Avijit Das, Lecturer of the department of Electrical and Electronic Engineering of BRAC University, for laying the foundation of our thesis project and providing valuable insight and guidance at each and every step of the development process. We are extremely fortunate and grateful to be able to work under his direct supervision. We want to thank him for continuous belief on us and frequent motivation to complete our project in time.

This is the project of Md Nafiz Ahmed, Aminul Islam, Toaha Hasnain, Iffat Tabassum, students from the Electrical and Electronic Engineering department of BRAC University. We would like to express our gratitude to the Almighty who gave us strength, determination, intelligence, help and opportunity to complete this project. We would also like to thank our family and friends for their continuous support and encouragement throughout the whole process of working for so long. Their cooperation made us becoming determined to achieve our goals. Last of all, our gratitude goes towards the faculty members of Department of Electrical and Electronic Engineering, BRAC University who provided us all of necessary and essential knowledge, concepts, ideas, support and tools that aided us to successfully complete our thesis project.

ABSTRACT

A license plate detect system is one type of intelligent transportation system. It is a type of technology in which the software enables computer system to read automatically the license number plate of vehicle from digital pictures. Reading automatically the number plate means converting the pixel information of digital image into the ASCII text of the number plate. The main objective is to use same operations repeatedly in such a way that the number plate of vehicle can be identified accurately.

As the problem of urban traffic congestion spreads, there is a pressing need for the introduction of advanced and to the technology equipment improve state of the control traffic nowadays and traffic problem increasing because the growing of are vehicles and we also have limited resource providing by current infrastructure. The simplest way to solve it is traffic monitoring system.

Another way is to use electronic sensors in order to detect vehicles, and produce signal that cycles. We propose a system for controlling the traffic light by image processing. The system will detect vehicles through images instead of using electronic sensors embedded in the pavement. A camera will be installed alongside the traffic light. It will capture image sequences.

Keywords: Roads/Highway, Automation, GPS, Image Processing, Communication.

•

CONTENTS

ABBREVIATIONS

1. **Inland water transport (IWT)**

2. **Dhaka Transport Planning and Coordination Board (GDTPCB)**

3. **Gaussian Mixture Model (GMM)**

4. **Automatic license plate recognition (ALPR)**

CHAPTER 1: INTRODUCTION

The application of image processing and computer vision techniques to the analysis of video sequences of traffic flow offers considerable improvements over the existing methods of traffic data collection and road traffic monitoring. Existing methods include the detectors such as loop, radar, infrared, ultrasonic, and microwave detectors which are expensive with limited capacity and involve installation, maintenance, and implementation difficulties. Image processing offer a relatively low installation cost with little traffic disruption during maintenance. Also they provide wide area monitoring allowing analysis of traffic flows and turning movements, speed measurement, multiple-point vehicle counts, vehicle classification and highway state assessment (e.g. congestion or incident detection). Image processing also finds extensive applications in the related field of autonomous vehicle guidance, mainly for determining the vehicle's relative position in the lane and for obstacle detection.

1.1 Roadway System of Bangladesh:

The Roadways system of Bangladesh consists of roads, railways, inland waterways, two Seaports, maritime shipping and civil aviation catering for both domestic and international traffic. Development and maintenance of transport infrastructure in the county is essentially the responsibilities of the public sector. The public sector is involved in transport operations in road, inland water transport (IWT) and ocean shipping alongside the private sector. In the road transport and IWT sub-sectors, the private sector is dominant. In ocean shipping, however, public sector still predominates, although the private sector has considerably increased its role in this sector in recent years. Recently private sector has also been involved in domestic air transport and railway in a very limited scale. Bangladesh witnessed rapid growth of transport since independence. The overall annual growth rate has been nearly 8.2 per cent for freight transport and 8.4 per cent for passenger transport. Even then, the transport intensity of the Bangladesh economy is considerably lower than that of many developing countries.

1.2 Problems in the Existing System:

Transport System Not Fully Integrated

Integrated system development which has now become a major issue in modern sustainable transport development, has particular significance for Bangladesh with her acute resource scarcity. Thus there is an urgent need for an optimum mix of modes and minimization of consumption of resources. However, such a mix cannot be achieved if one looks at a mode in isolation from others. Thus although rail and water transport is generally more efficient than road transport because of their higher energy efficiency and better labor productivity, this fact by itself cannot ensure greater use of these modes. In most of the cases they alone cannot provide door-to-door services. Because of their higher terminal costs they are also not suitable for short trip length or where intensity of demand is too low to justify higher capacity modes. These inherent characteristics of different modes require that to improve overall efficiency each mode should be used for what it does best in an overall transport chain. Reflecting a fundamental change in the traditional way of looking at transportation of goods and people, a mode is increasingly considered only as a link in the chain and the whole issue of transportation from the origin to ultimate destination is considered. In Bangladesh, each mode of transport

operators on its own without any initiative to establish efficient logistic chains between O-D involving different modes as necessary. Thus an integrated system involving different modes, as appropriate from the origin to ultimate destination is needed.

Lack of Transport Policy

Bangladesh has no transport policy as yet. As such there is no clear decision as to which modes of transport and facilities, the urban areas should encourage. In the past urban transport received little attention, as investment went more in infrastructure development for inter-urban linkages and for opening up links to rural growth centers. The 4th Five Year Plan of Bangladesh (1990-95) indicated that urban transport problems will be tackled, particularly in the metropolitan areas with emphasis on land use and water management system.

Government therefore, undertook a study "the Greater Dhaka Metropolitan Area Integrated Transport Study (DITS) (1992-94), funded by UNDP. In line with the findings of the study, World Bank formulated a project – "Dhaka Urban Transport Project", to address in the short-term, urgent policy issues, infrastructure bottlenecks and traffic management constraints, and in the longer term, to focus on planning, institutional and policy action. Based on another recommendation of the World Bank for strengthening coordination mechanism, Greater Dhaka Transport Planning and Coordination Board (GDTPCB) were established. The Board has recently been renamed as Dhaka Transport Coordination Board. While efforts are underway to improve urban transport situation in Dhaka, similar initiatives need to be taken to address urban transport problems in other cities, and before that there is an urgent need for setting urban transport policies of Bangladesh.

Lack of Security

The vehicle security of Bangladesh is decreasing day by day significantly. Some automation system may be introduced so that the security system of Bangladesh vehicle may increase. For that reason License plate detection method is the most acceptable method which will create vast revolution in the vehicle security system in Bangladesh.

1.3 Objective:

Our aim is to decrease the traffic congestion and increase the security of Bangladesh by the application of Image Processing in Traffic Control System and License Plate Detection.

1.4 Project Overview:

We have introduced Gaussian mixture model to detect the foreground and background of a video frame.

Here ForegroundDetector method takes 4 parameters. First one is to declare the model we have used.Number of Gaussian modes in the mixture model, specified as the comma-separated pair consisting of 'NumGaussians' and a positive integer. Typically this value is 3, 4 or 5. Set this value to 3 or greater to be able to model multiple background modes.

Last two parameters are to specify the number of training frames to detect the foreground.

- videoReader = vision.VideoFileReader('MVI_6762.MOV');

This function is used to declare a videoReader object to read frame by frame of a particular video.

- frame = step(videoReader);

This step function reads one frame at a time from the whole video.

- foreground = step(foregroundDetector, frame);

Here step function is used to get the only the foreground of an image.

- Next commented region was used to omit the unnecessary regions of the image. Like parts of the image which are not part of the main road can be omitted through these commented lines.
- se = strel('square', 5);

This function is used to declare a structural object to do morphological operations. Here we have used a squared shaped structural object with a dimension of 5*5.

- filteredForeground = imopen(foreground, se);

This function does the morphological operation ‘opening’. This is a procedure of doing erosion followed by one dilation. This function is used to remove noises from an image. It also disconnects objects which are connected through few number of pixels. So that unique objects are detected separately.

- blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, 'AreaOutputPort', true, 'CentroidOutputPort', true, 'MinimumBlobArea', 3000);

This function is used to count the number of objects of an image. Here we are detecting objects with minimal area of 3000 pixels.

- bbox = step(blobAnalysis, filteredForeground);

Finally this step function is used to get the objects in the foreground image.

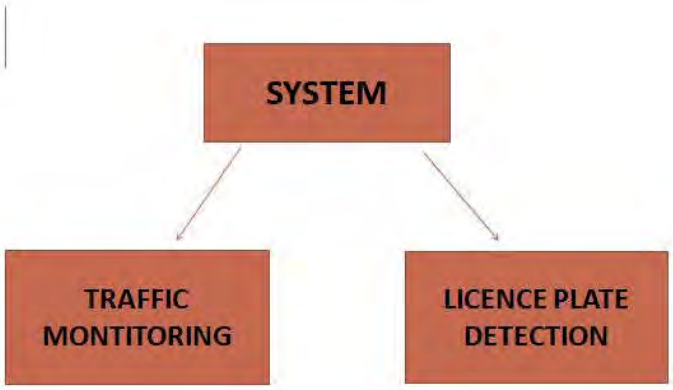
- result = insertShape(mainFrame, 'Rectangle', bbox, 'Color', 'green');

This functions is used to insert or mark the detected objects with a green colored rectangle box.

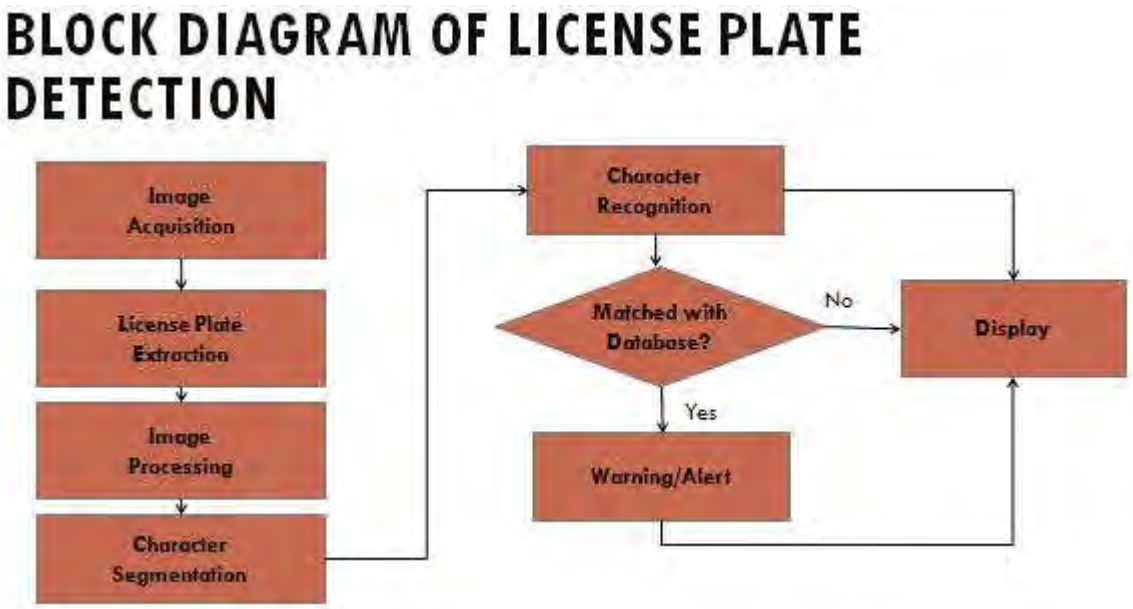
- for i=1:r

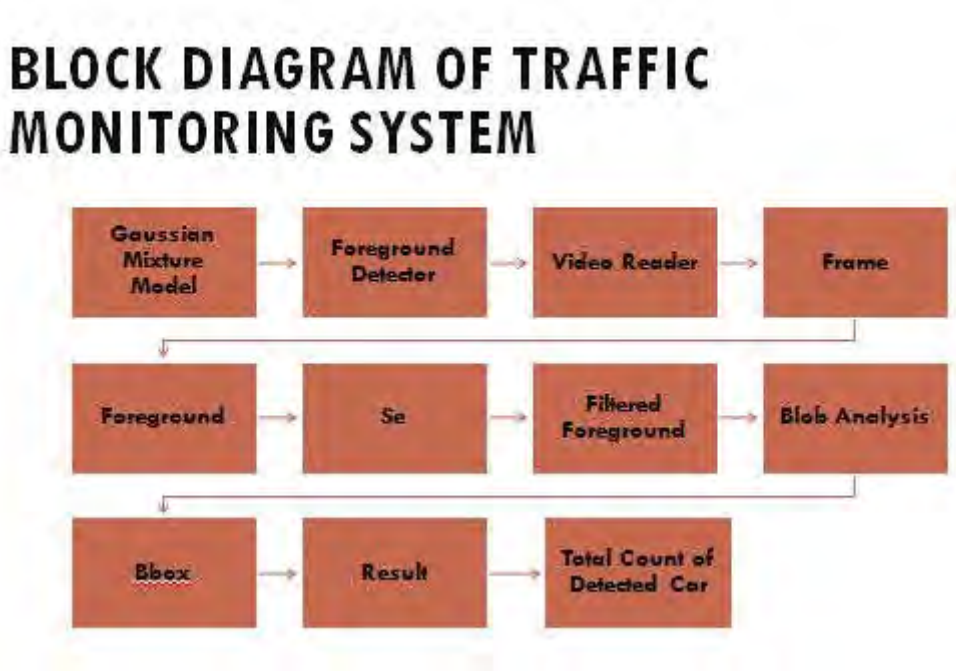
```
if (centroids(i,2) > 676 && centroids(i,2) < 680 )
        numCars = numCars + 1;
end
end
```

I have declared a small region to count the number of cars of an image. We are tracking the centroid of a detected object. Whenever the detected centroid is crossing that particular region we are counting one vehicle. This region must be very small so that one vehicle is not being counted twice.



CHAPTER: 2: SYSTEM DESCRIPTION





2.1 Application of Image Processing License Plate Detection [SYSTEM DESCRIPTION]

Overview:

In license plate detection we have used image processing method with raspberry pi. It is mainly a process to detect the number plate of a vehicle. In our country it will be very beneficial to detect the number plate. Our motivation was we were tried to do work in large sector with it and do the implementation in field. We have used the coding language is python and the raspberry pi which model is 3B. We have used the alpr method to detect the number plate.

Raspberry pi:

Here we have used the main signal processing chip unit used in Raspberry Pi 3B model. The system is a Broadcom 2835 700MHz Chip in which CPU core is a 32 bit RISC processor designed by Advanced RISC Machines. In this main processing chip connects a camera and display. The Raspberry Pi design does not include a built in hard disk or solid state drive. We used an SD card for booting and long term storage. The board is intended to run Linux which is based on operating systems. This Raspberry Pi has a Samsung class 32 GB micro SD card, which is preloaded with the Raspberry Pi NOOBS (New Out of Box Software) package, and a printed Micro SD card adaptor. There are several types of connectors in this Raspberry PI. There is an Ethernet connector with 10/100 BaseT Ethernet socket. There is video output which is HDMI (rev 1.3 & 1.4 Composite RCA (PAL and NTSC). It also has an audio output, the configuration of which is Audio Output 3.5mm jack, HDMI USB 4 x USB 2.0 Connector. There is GPIO connector with 40-pin 2.54 mm (100 mil) expansion header: 2x20 strip Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines configurations. This Raspberry PI's camera connector is of 15-pin MIPI Camera Serial Interface (CSI-2) configuration. It has a display connector which has Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane configuration. Advantages of this model of Raspberry PI are that it is low cost and it has a consistent board format. Moreover, this model has ten times faster processing speed than previous models. Last but not the least, this model of Raspberry PI has added connectivity.

Fig1: Raspberry pi 3B (board, front view)

Basic Concept of image processing:

It generally a basic concept of image processing. It consists of three stages. Input, processor, output. First of all it captured by a camera .Then it send to a particular process of pixel then it processed to a output.

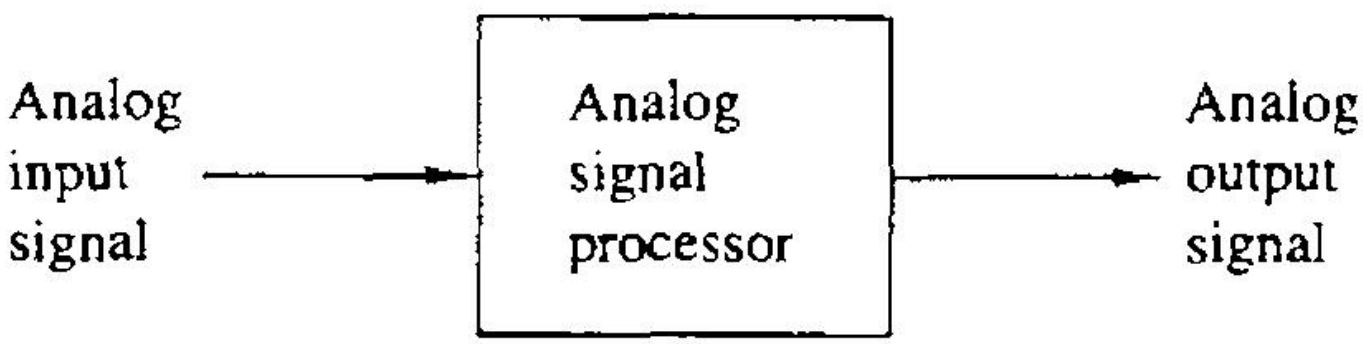


Fig: simple block diagram of image processing.

Camera Interface:

In this project we have used a camera which is 8 Mega pixels. It is basically a camera of a mobile phone. We have used it here just for experiment. The camera module is connected is for the CSI connector with the raspberry pi. It is able to delivery 1080p HD resolution with 30fps. The camera module is attached to the raspberry pi with a 15 pin cable. It is specially design for the raspberry pi system. The CSI is designed to pass the high rated data and a high pixel photo. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the BCM2835 processor.

It has an image sensor which is Omnivision 5647 CMOS image sensor in a fixed-focus module with integral IR filter. It has still picture resolution of 2592 x 1944. It gets connected to the Raspberry PI via 15 Pin ribbon cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI-2). It has image control functions such as Automatic exposure control, Automatic white balance, Automatic band filter, Automatic 50/60 Hz luminance detection and Automatic black level calibration. It has an operating temperature range from -30˚ to 70˚. This camera's lens size is 1/4". This camera module weighs 3g and has a dimension of 20 x 25 x 10mm.

Fig2: Raspberry pi model 3B.

Methodology:

The method we have used here is basically the alpr method. It is means that automatic license plate recognition. It is a huge big file of python coding system. This system is mainly designed for number plate recognition. Automatic License Plate Recognition system is basically a real time embedded system that is automatically recognizes the license plate of cars and vehicles. It has many applications in complex security systems. For common areas and from parking admission to urban traffic control. Automatic license plate recognition (ALPR) has complex characteristics due to diverse effects such as of light and speed. We have used here the python library. ALPR is basically doing the image processing.

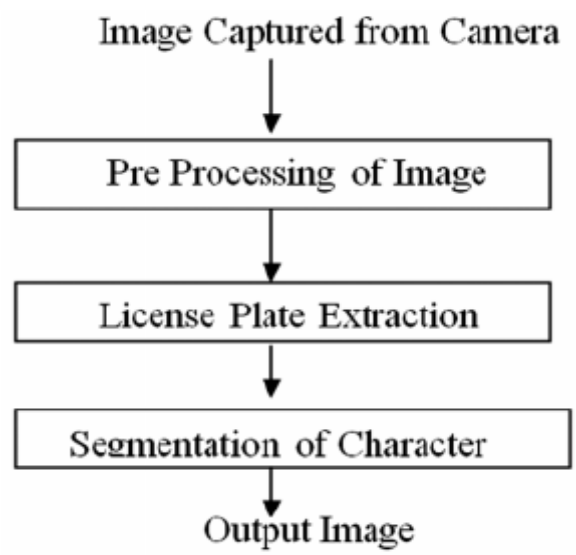


Fig: Block diagram of the system.

Automatic license plate recognition: (ALPR):

ALPR means automatic license plate recognition. The scientific works has reached so high that they have invented a new technology. The technology is the alpr system. Basically it has introduced firstly at the USA. The police of the country first use this for a trial. Now it has become very beneficial to them. By this system we can use it in particular motivation. We can use t for example for finding the stolen car. In our country car stealing is a very common phenomenon. It is very sad that we cannot do anything for this. Our motivation to do this project was we will try to do the implementation of this system. It will be very good if we can apply it in our traffic system. Basically it's costs a very cheap rate. If we have a good camera then we can apply this on our traffic system. It will record all the data of the vehicles license plate.

Automatic license plate recognition is a automatic embedded system that can recognize the license plate by itself. It is a complex security system to common areas. It's a security system of parking and traffic controlling system too. It has complex character and diverse features .It has control over lights and speed. Most of the ALPR system is designed by MATLAB and python. It's easier to do work through python software. In our project we have used the python coding language. ALPR is a computer vision technology that extracts the image from the license plate. It's specially design for recognition of license plate. Then it do the extraction of the image by the image processing system. After image processing it do the character segmentation. Here it basically identify the character from the license plate. It removes the extra pictures or sides from the license plate. It only focus on the characters of the license plate. Then it do the segmentation of the character of its own process. Then we get the result or the output on the display. In our project we can see the output in a LINUX screen. There we can see a row for the plate number and others for the confidence. The confidence is about how much it has matched with the binary possible number or character.

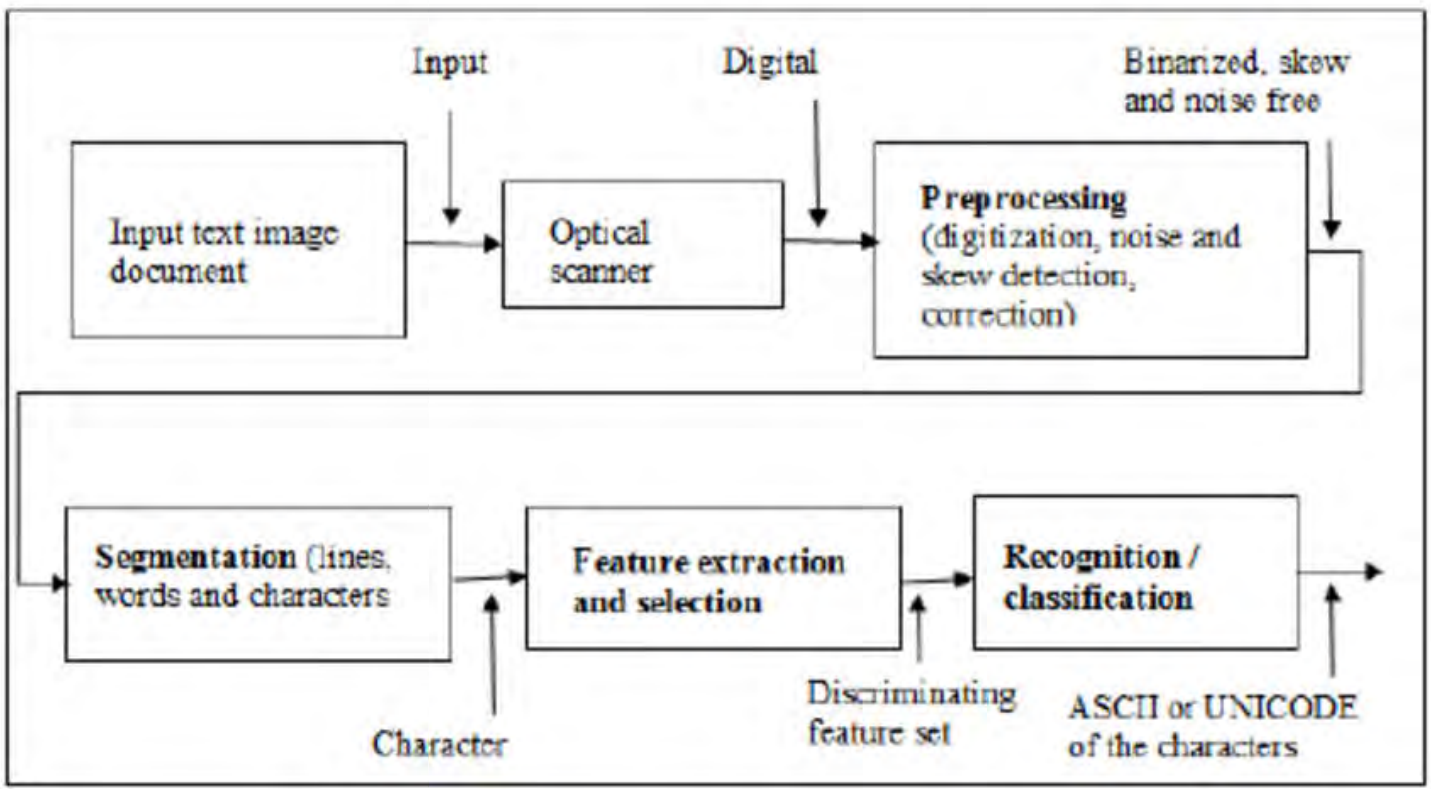


Fig: block diagram of ALPR system.(details version)

Basic programme of the ALPR system:

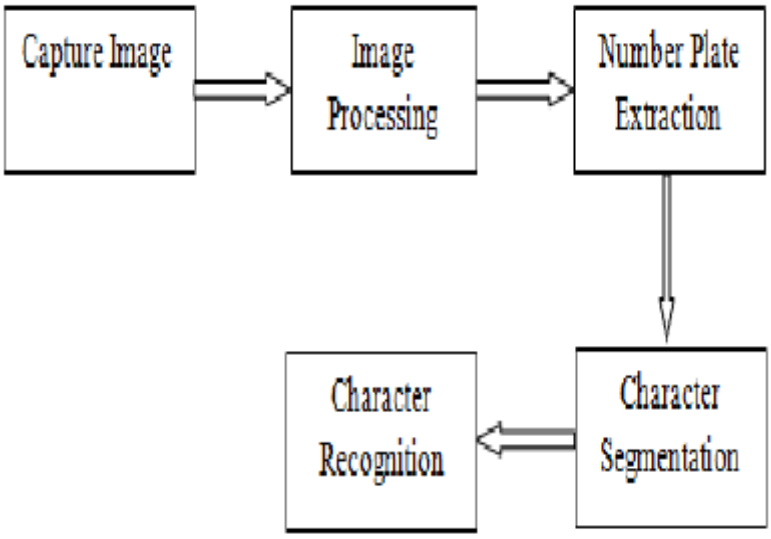


Fig: Basic block diagram of the ALPR system

2.2 Application of Image Processing in Traffic Control System

[SYSTEM DESCRIPTION]

A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMMs are commonly used as a parametric model of the probability distribution of continuous measurements or features in a biometric system, such as color based tracking of an object in video. In many computer related vision technology, it is critical to identify moving objects from a sequence of videos frames. In order to achieve this, background subtraction is applied which mainly identifies moving objects from each portion of video frames. Background subtraction or segmentation is a widely used technique in video surveillance, target recognitions and banks. By using the Gaussian Mixture Model background model, frame pixels are deleted from the required video to achieve the desired results. The application of background subtraction involves various factors which involve developing an algorithm which is able to detect the required object robustly, it should also be able to react to various changes like illumination, starting and stopping of moving objects. Surveillance is the monitoring of the behavior, activities or other changing information usually of people and often in a cast by moving objects are at times mistaken for true foreground objects.

Algorithm of Gaussian Mixture Model:

In order to give a better understanding of the algorithm used for background subtraction the following steps were adopted to achieve the desired results:
1. Firstly, we compare each input pixels to the mean 'μ' of the associated components. If the value of a pixel is close enough to a chosen component's mean, then that component is considered as the matched component. In order to be a matched component, the difference between the pixels and mean must be less than compared to the component's standard deviation scaled by factor D in the algorithm. 2. Secondly, update the Gaussian weight, mean and standard deviation (variance) to reflect the new obtained pixel value. In relation to non-matched components the weights 'w' decreases whereas the mean and standard deviation stay the same. It is dependent upon the learning component 'p' in relation to how fast they change.

3. Thirdly, here we identify which components are parts of the background model. To do this a threshold value is applied to the component weights 'w'.
4. Fourthly, in the final step we determine the foreground pixels. Here the pixels that are identified as foreground don't match with any components determined to be the background.

General formula of Gaussian Mixture Model:

A Gaussian mixture model can be formulated in general as follows:

$$\mu_t = \sum_{i=1}^{K} \omega_{i,t}\mu_{i,t}$$

Were obviously,

$$\sum_{i=1}^{K} \omega_{i,t} = 1$$

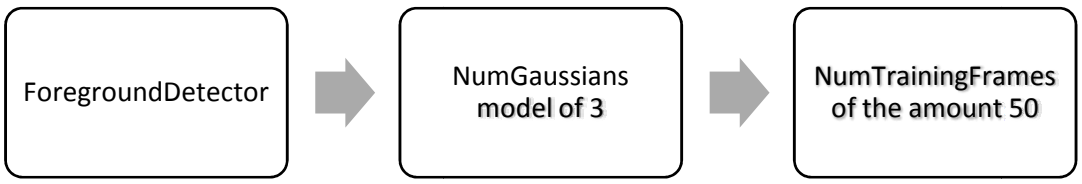
The mean of such a mixture equals

$$\mu_t = \sum_{i=1}^{K} \omega_{i,t}\mu_{i,t}$$

that is, the weighted sum of the means of the component densities. Where the variable which represents the current pixel in frame, K be is the number of distributions, and t represents time
(i.e., the frame index), is an estimate of the weight of the ith Gaussian in the mixture at time t, is the mean value of the ith Gaussian in the mixture at time t, is the covariance matrix
of the ith Gaussian in the mixture at time t.

Code Algorithm:

vision.ForegroundDetector



Description:

The ForegroundDetector System object compares a color or grayscale video frame to a background model to determine whether individual pixels are part of the background or the foreground. It then computes a foreground mask. By using background subtraction, you can detect foreground objects in an image taken from a stationary camera.

Construction:

detector = vision.ForegroundDetector returns a foreground detector System object, detector. Given a series of either grayscale or color video frames, the object computes and return the foreground mask using Gaussian mixture models (GMM).

Properties:

NumTrainingFrames — Number of initial video frames for training background model
150 (default) | integer

Number of initial video frames for training background model, specified as the comma-separated pair consisting of 'NumTrainingFrames' and an integer. When you set the AdaptLearningRate to false, this property will not be available.

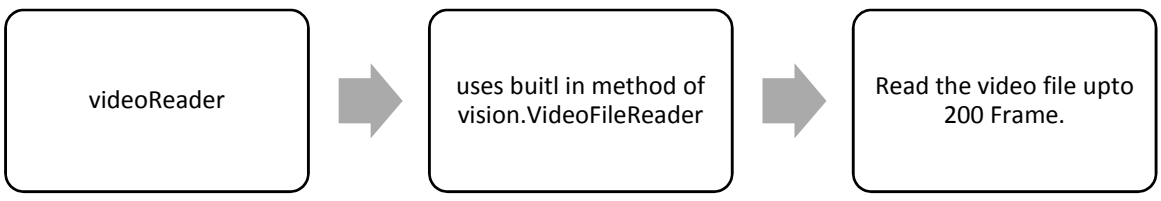
Syntax:

```
foregroundDetector = vision.ForegroundDetector('NumGaussians', 3, ...
    'NumTrainingFrames', 50);
```

Results:

```
NumGaussians: 3
MinimumBackgroundRatio: 0.7000
InitialVariance: 'Auto'
AdaptLearningRate: true
NumTrainingFrames: 50
LearningRate: 0.0050
```

vision.VideoFileReader('MVI_6762.MOV');



Description:

The VideoFileReader object reads video frames, images, and audio samples from a video file. The object can also read image files.

Supported Platforms and File Types

The supported file formats available to you depend on the codecs installed on your system.

Platform	Supported File Name Extensions
All Platforms	AVI (.avi)
Windows®	**Image**: .jpg,.bmp
	Video: MPEG (.mpeg) MPEG-2 (.mp2) MPEG-1.mpg MPEG-4, including H.264 encoded video (.mp4, .m4v) Motion JPEG 2000 (.mj2) Windows Media Video (.wmv,.asf, .asx, .asx) and any format supported by Microsoft DirectShow® 9.0 or higher.
	Audio: WAVE (.wav) Windows Media Audio File (.wma) Audio Interchange File Format (.aif, .aiff) Compressed Audio Interchange File Format(.aifc), MP3 (.mp3) Sun Audio (.au) Apple (.snd)
Macintosh	**Video**: .avi

Platform	Supported File Name Extensions
	Motion JPEG 2000 (.mj2) MPEG-4, including H.264 encoded video (.mp4, .m4v) Apple QuickTime Movie (.mov) and any format supported by QuickTime as listed on http://support.apple.com/kb/HT3775.
	Audio: Uncompressed .avi
Linux®	Motion JPEG 2000 (.mj2) Any format supported by your installed plug-ins for GStreamer 0.1 or higher, as listed on http://gstreamer.freedesktop.org/documentation/plugins.html, including Ogg Theora (.ogg).

Construction:

videoFReader = vision.VideoFileReader(Filename) returns a video file reader System object, videoFReader. The object can sequentially read video frames and/or audio samples from the input video file, FILENAME. Every call to the step method returns the next video frame.

videoFReader = vision.VideoFileReader(Filename,*Name*,*Value*) configures the video file reader properties, specified as one or more name-value pair arguments. Unspecified properties have default values.

To read a file:
Define and set up your video file reader object using the constructor.
Call the step method with the input filename, FILENAME, the video file reader object, videoFReader, and any optional properties. See the syntax below for using the step method.
I = step(videoFReader) outputs next video frame.
[Y,Cb,Cr] = step(videoFReader) outputs the next frame of YCbCr 4:2:2 format video in the color components Y, Cb, and Cr. This syntax requires that you set the ImageColorSpace property to 'YCbCr 4:2:2'.
[I,AUDIO] = step(videoFReader) outputs the next video frame, I, and one frame of audio samples, AUDIO. This syntax requires that you set the AudioOutputPort property to true.
[Y,Cb,Cr,AUDIO] = step(videoFReader) outputs next frame of YCbCr 4:2:2 video in the color components Y, Cb, and Cr, and one frame of audio samples in AUDIO. This syntax requires that you set the AudioOutputPort property to true, and the ImageColorSpace property to 'YCbCr 4:2:2'.
[..., EOF] = step(videoFReader) returns the end-of-file indicator, EOF. The object sets EOF to true each time the output contains the last audio sample and/or video frame.

Properties:

Filename — Name of video file

Specify the name of the video file as a character vector. The full path for the file needs to be specified only if the file is not on the MATLAB® path.

Default: vipmen.avi

Syntax:

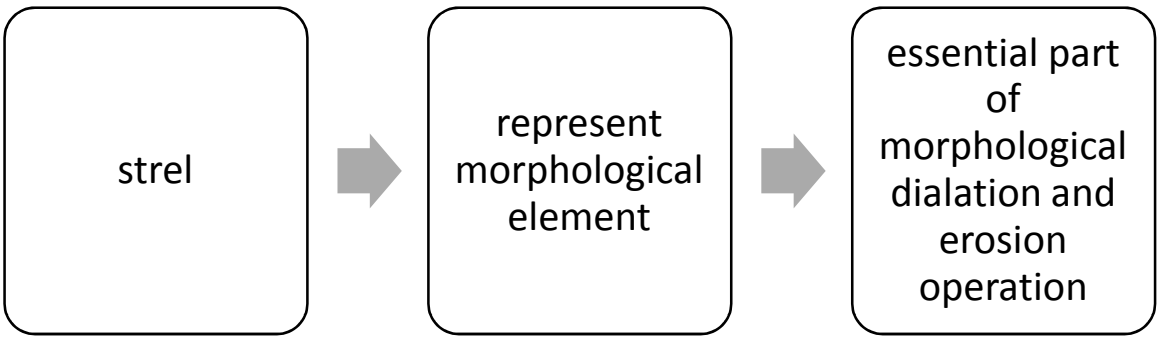
```
videoReader = vision.VideoFileReader('MVI_6762.MOV');

for i = 1:200
%     frame = read(v,i); % read the next video frame
    frame = step(videoReader);
%     h = ones(3,3)/25;
%     frame = imfilter(frame,h);
    foreground = step(foregroundDetector, frame);
end
```

Result:

Video Properties:
Width: 1280
Height: 720
FrameRate: 59.9401
BitsPerPixel: 24
VideoFormat: 'RGB24'

strel:

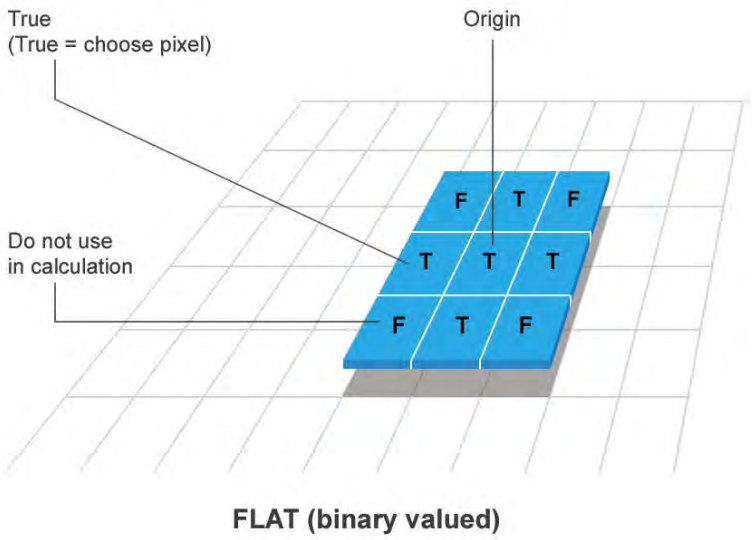


Create morphological structuring element.

Structuring Elements:

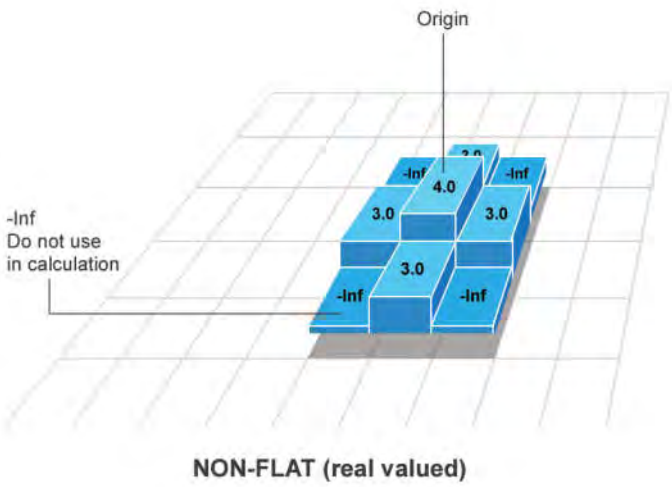
An essential part of the morphological dilation and erosion operations is the structuring element used to probe the input image. A structuring element is a matrix that identifies the pixel in the image being processed and defines the neighborhood used in the processing of each pixel. You typically choose a structuring element the same size and shape as the objects you want to process in the input image. For example, to find lines in an image, create a linear structuring element.

There are two types of structuring elements: *flat* and *nonflat*. A flat structuring element is a binary valued neighborhood, either 2-D or multidimensional, in which the true pixels are included in the morphological computation, and the false pixels are not. The center pixel of the structuring element, called the *origin*, identifies the pixel in the image being processed. Use the strel function to create a flat structuring element. You can use flat structuring elements with both binary and grayscale images. The following figure illustrates a flat structuring element.



FLAT (binary valued)

A nonflat structuring element is a matrix of type double that identifies the pixel in the image being processed and defines the neighborhood used in the processing of that pixel. A nonflat structuring element contains finite values used as additive offsets in the morphological computation. The center pixel of the matrix, called the *origin*, identifies the pixel in the image that is being processed. Pixels in the neighborhood with the value -Inf are not used in the computation. Use the offsetstrel function to create a nonflat structuring element. You can use nonflat structuring elements only with grayscale images.



NON-FLAT (real valued)

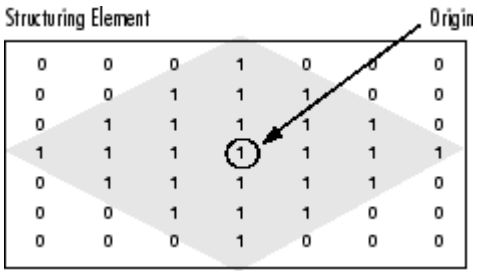
Determine the Origin of a Structuring Element:

The morphological functions use this code to get the coordinates of the origin of structuring elements of any size and dimension:

origin = floor((size(nhood)+1)/2)

where nhood is the neighborhood defining the structuring element. To see the neighborhood of a flat structuring element, view the Neighborhood property of the strel object. To see the neighborhood of a nonflat structuring element, view the Offset property of the offsetstrel object.

For example, the following illustrates the origin of a flat, diamond-shaped structuring element.



Structuring Element Decomposition:

To enhance performance, the strel and offsetstrel functions might break structuring elements into smaller pieces, a technique known as *structuring element decomposition*.

For example, dilation by an 11-by-11 square structuring element can be accomplished by dilating first with a 1-by-11 structuring element, and then with an 11-by-1 structuring element. This results in a theoretical speed improvement of a factor of 5.5, although in practice the actual speed improvement is somewhat less.

Structuring element decompositions used for the 'disk' and 'ball' shapes are approximations; all other decompositions are exact. Decomposition is not used with an arbitrary structuring element unless it is a flat structuring element whose neighborhood matrix is all 1's.

To see the sequence of structuring elements used in a decomposition, use the decompose method. Both strel objects and offsetstrel objects support decompose methods. The decompose method returns an array of the structuring elements that form the decomposition. For example, here are the structuring elements created in the decomposition of a diamond-shaped structuring element.

Construction:

SE = strel('square',W) creates a square structuring element whose width is W pixels.

Example:

SE = strel('square', 11)

SE =
strel is a square shaped structuring element with properties:

Neighborhood: [11×11 logical]
Dimensionality: 2

Properties:

Neighborhood — structuring element neighborhood logical matrix

Structuring element neighborhood, specified as a matrix of class logical.

Data Types: logical

Dimensionality — Dimensions of structuring element nonnegative scalar

Dimensions of structuring element, specified as a nonnegative scalar of class double.

Data Types: double

Algorithms:

For all shapes except 'arbitrary', structuring elements are constructed using a family of techniques known collectively as *structuring element decomposition*. The principle is that dilation by some large structuring elements can be computed faster by dilation with a sequence of smaller structuring elements. For example, dilation by an 11-by-11 square structuring element can be accomplished by dilating first with a 1-by-11 structuring element and then with an 11-by-1 structuring element. This results in a theoretical performance improvement of a factor of 5.5, although in practice the actual performance improvement is somewhat less.

Imopen

Morphologically open image

Description:

IM2 = imopen(IM,SE) performs morphological opening on the grayscale or binary image IM with the structuring element SE. The argument SE must be a single structuring element object, as opposed to an array of objects. The morphological open operation is an erosion followed by a dilation, using the same structuring element for both operations.

IM2 = imopen(IM,NHOOD) performs opening with the structuring element strel(NHOOD), where NHOOD is an array of 0's and 1's that specifies the structuring element neighborhood.

gpuarrayIM2 = imopen(gpuarrayIM,___) performs the operation on a graphics processing unit (GPU) with the structuring element strel(NHOOD), if NHOOD is an array of 0s and 1s that specifies the structuring element neighborhood, or strel(gather(NHOOD)) if NHOOD is a gpuArray object that specifies the structuring element neighborhood. This syntax requires the Parallel Computing Toolbox™.

Class Support:

IM can be any numeric or logical class and any dimension, and must be nonsparse. If IM is logical, then SE must be flat.

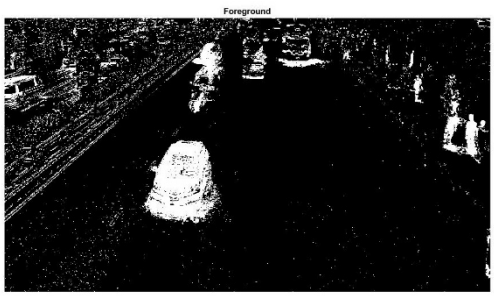
gpuarrayIM must be a gpuArray of type uint8 or logical. When used with a gpuarray, the structuring element must be flat and two-dimensional.

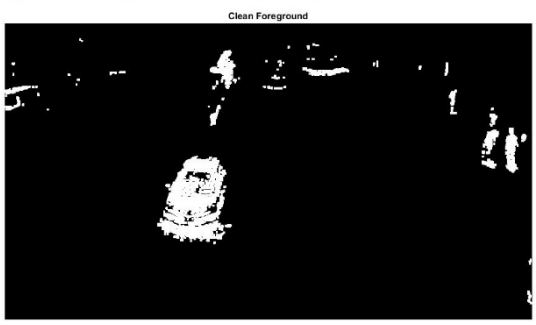
The output has the same class as the input.

Syntax:

```
se = strel('square', 5);
filteredForeground = imopen(foreground, se);
%figure; imshow(filteredForeground); title('Clean Foreground');
```

Result:





Step

step response is plot of dynamic system which is actually step response data. This step function is used to get the objects in the foreground image.

Syntax

```
step(sys)
step(sys,Tfinal)
step(sys,t)
step(sys1,sys2,...,sysN)
step(sys1,sys2,...,sysN,Tfinal)
step(sys1,sys2,...,sysN,t)
y = step(sys,t)
[y,t] = step(sys)
[y,t] = step(sys,Tfinal)
[y,t,x] = step(sys)
[y,t,x,ysd] = step(sys)
[y,...] = step(sys,...,options)
```

Description

step analyses the step response of a dynamic system. For the state-space case, zero initial state is assumed. When it is invoked with no output arguments, this function plots the step response on the screen.

step(sys) plots the step response of an arbitrary dynamic system model, sys. This model can be continuous- or discrete-time, and SISO or MIMO. The step response of multi-input systems is the collection of step responses for each input channel. The duration of simulation is determined automatically, based on the system poles and zeros.

step(sys,Tfinal) simulates the step response from t = 0 to the final time t = Tfinal. Express Tfinal in the system time units, specified in the TimeUnit property of sys. For discrete-time systems with unspecified sample time (Ts = -1), step interprets Tfinal as the number of sampling periods to simulate.

step(sys,t) uses the user-supplied time vector t for simulation. Express t in the system time units, specified in the TimeUnit property of sys. For discrete-time models, t should be of the form Ti:Ts:Tf, where Ts is the sample time. For continuous-time models, t should be of the form Ti:dt:Tf, where dt becomes the sample time of a discrete approximation to the continuous system .The step command always applies the step input at t=0, regardless of Ti.

To plot the step response of several models sys1,..., sysN on a single figure, use

step(sys1,sys2,...,sysN)

step(sys1,sys2,...,sysN,Tfinal)

step(sys1,sys2,...,sysN,t)

All of the systems plotted on a single plot must have the same number of inputs and outputs. However, we can plot a mix of continuous- and discrete-time systems on a single plot. This syntax is useful to compare the step responses of multiple systems.

We can also specify a distinctive color, linestyle, marker, or all three for each system. For example,

step(sys1,'y:',sys2,'g--')

plots the step response of sys1 with a dotted yellow line and the step response of sys2 with a green dashed line.

When invoked with output arguments:

y = step(sys,t)

[y,t] = step(sys)

[y,t] = step(sys,Tfinal)

[y,t,x] = step(sys)

step returns the output response y, the time vector t used for simulation (if not supplied as an input argument), and the state trajectories x (for state-space models only). No plot generates on the screen. For single-input systems, y has as many rows as time samples (length of t), and as many columns as outputs. In the multi-input case, the step responses of each input channel are stacked up along the third dimension of y. The dimensions of y are then

(length of t)×(number of outputs)×(number of inputs)

and y(:,:,j) gives the response to a unit step command injected in the jth input channel. Similarly, the dimensions of x are

(length of t)×(number of states)×(number of inputs)

For identified models (see idlti and idnlmodlel) [y,t,x,ysd] = step(sys) also computes the standard deviation ysd of the response y (ysd is empty if sys does not contain parameter covariance information).

[y,...] = step(sys,...,options) specifies additional options for computing the step response, such as the step amplitude or input offset. Use stepdataoptions create the option set options.

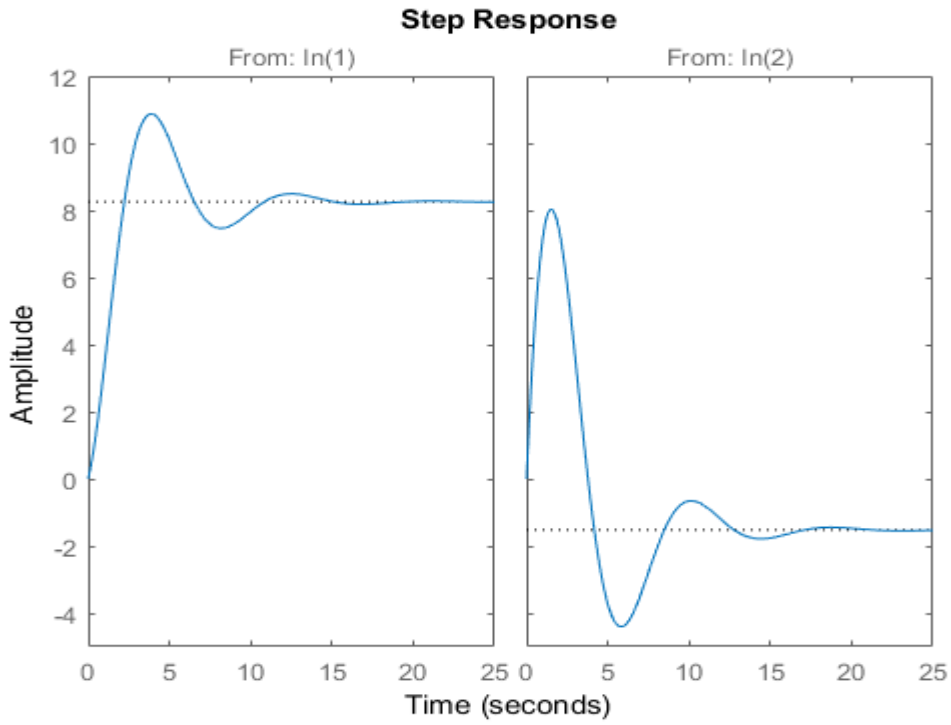
Examples

Plot the step response of the following second-order state-space model:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -0.5572 & -0.7814 \\ 0.7814 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$y = \begin{bmatrix} 1.9691 & 6.4493 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

```
a = [-0.5572,-0.7814;0.7814,0];
b = [1,-1;0,2];
c = [1.9691,6.4493];
sys = ss(a,b,c,0);
step(sys)
```



The left plot shows the step response for first input and second plot shows the step response for second input.

InsertShape

'insertShape' is used to insert shapes in an image or video.

Syntax

RGB = insertShape(I,shape,position)
RGB = insertShape(___,Name,Value)

Description

`RGB = insertShape(I,shape,position)` returns a truecolor image with shape inserted. The input image, `I`, can be either a truecolor or grayscale image. We draw the shapes by overwriting pixel values.

`RGB = insertShape(___,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments.

Examples

I. **Insert Circle and Filled Shapes on an Image**

Read the image.

I = imread('peppers.png');

Draw a circle with a border line width of 5.

```
RGB = insertShape(I,'circle',[150 280 35],'LineWidth',5);
```

Draw a filled triangle and a filled hexagon.

```
pos_triangle = [183 297 302 250 316 297];
pos_hexagon = [340 163 305 186 303 257 334 294 362 255 361 191];
RGB = insertShape(RGB,'FilledPolygon',{pos_triangle,pos_hexagon},...
    'Color', {'white','green'},'Opacity',0.7);
```

Display the image.

```
imshow(RGB);
```

II. Input Arguments

a.I — Input image

M-by-*N*-by-3 truecolor | *M*-by-*N* 2-D grayscale image

Input image, specified in truecolor or 2-D grayscale.

Data Types: single | double | int16 | uint8 | uint16

b. shape — Type of shape

character vector

Type of shape, specified as a character vector. The vector can be, 'Rectangle', 'FilledRectangle', 'Line', 'Polygon', 'FilledPolygon', 'Circle', or 'FilledCircle'.

Data Types: char

c. position — Position of shape

matrix | vector | cell array

Position of shape, specified according to the type of shape, described in the table.

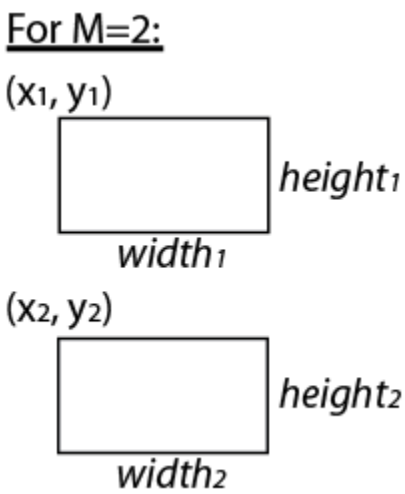
Shape

'Rectangle'

'FilledRectangle'

M-by-4 matrix where each row specifies a rectangle as [x y width height]

$$\begin{bmatrix} x_1 & y_1 & width_1 & height_1 \\ x_2 & y_2 & width_2 & height_2 \\ \vdots & \vdots & \vdots & \vdots \\ x_M & y_M & width_M & height_M \end{bmatrix}$$



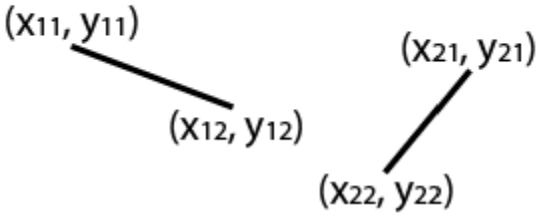
Shape

'Line'

For one or more disconnected lines, an *M*-by-4 matrix, where each four-element vector $[x_1, y_1, x_2, y_2]$, describe a line with endpoints, $[x_1\ y_1]$ and $[x_2\ y_2]$.

$$\begin{bmatrix} x_{11} & y_{11} & x_{12} & y_{12} \\ x_{21} & y_{21} & x_{22} & y_{22} \\ \vdots & \vdots & \vdots & \vdots \\ x_{M1} & y_{M1} & x_{M2} & x_{M2} \end{bmatrix}$$

For M=2:



For one or more line segments, an *M*-by-2*L* matrix, where each row is a vector representing a polyline with *L* number of vertices.

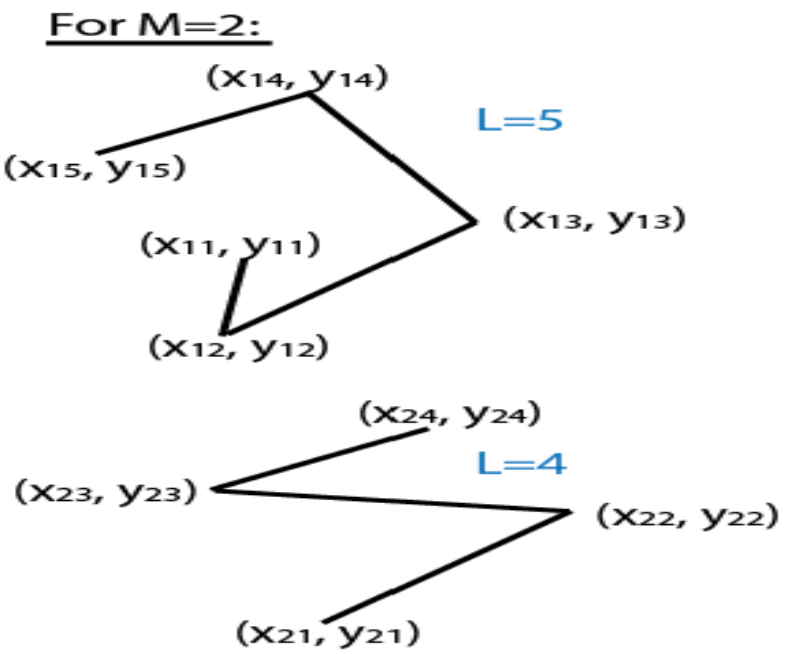
$$\begin{bmatrix} x_{11} & y_{11} & x_{12} & y_{12} & \cdots & x_{1L} & y_{1L} \\ x_{21} & y_{21} & x_{22} & y_{22} & \cdots & x_{2L} & y_{2L} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{M1} & y_{M1} & x_{M2} & y_{M2} & \cdots & x_{ML} & y_{ML} \end{bmatrix}$$

The polyline always contains (*L*-1) number of segments because the first and last vertex points do not connect. For lines with fewer segments, repeat the ending coordinates to fill the matrix.

You can also specify the shapes as a cell array of *M* vectors.

$\{[x_{11},y_{11},x_{12},y_{12},...,x_{1p},y_{1p}], [x_{21},y_{21},x_{22},y_{22},.....,x_{2q},y_{2q}], ... [x_{M1},y_{M1},x_{M2},y_{M2},.....,x_{Mr},y_{Mr}]\}$

p, *q*, and *r* specify the number of vertices.



Shape

'Polygon'
'FilledPolygon'

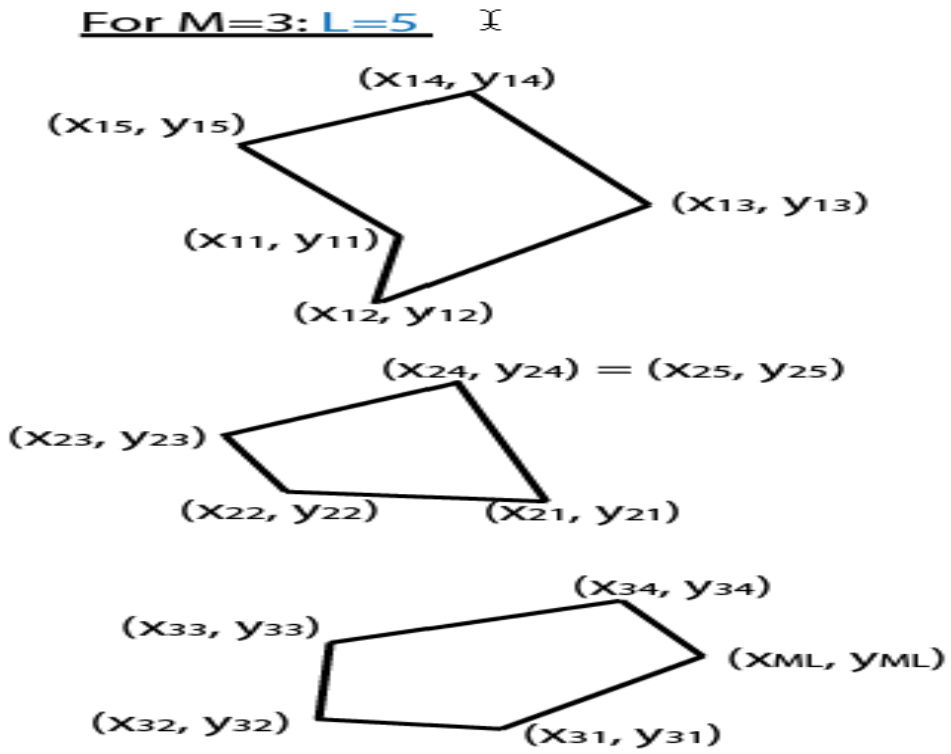
An *M*-by-2*L* matrix, where each row represents a polygon with *L* number of vertices. Each row of the matrix corresponds to a polygon. For polygons with fewer segments, repeat the ending coordinates to fill the matrix.

$$\begin{bmatrix} x_{11} & y_{11} & x_{12} & y_{12} & \cdots & x_{1L} & y_{1L} \\ x_{21} & y_{21} & x_{22} & y_{22} & \cdots & x_{2L} & y_{2L} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{M1} & y_{M1} & x_{M2} & y_{M2} & \cdots & x_{ML} & y_{ML} \end{bmatrix}$$

We can also specify the shapes as a cell array of *M* vectors:

$\{[x_{11},y_{11},x_{12},y_{12},...,x_{1p},y_{1p}], [x_{21},y_{21},x_{22},y_{22},.....,x_{2q},y_{2q}], ... [x_{M1},y_{M1},x_{M2},y_{M2},.....,x_{Mr},y_{Mr}]\}$

p, *q*, and *r* specify the number of vertices.



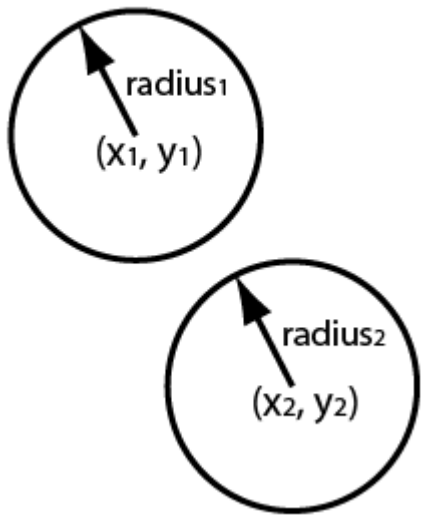
Shape

'Circle'
'FilledCircle'

An *M*-by-3 matrix, where each row is a vector specifying a circle as [x y radius]. The [x y] coordinates represent the center of the circle.

$$\begin{bmatrix} x_1 & y_1 & radius_1 \\ x_2 & y_2 & radius_2 \\ \vdots & \vdots & \vdots \\ x_M & y_M & radius_M \end{bmatrix}$$

For M=2:



Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

III. Name-Value Pair Arguments

Specify optional comma-separated pairs of Name,Value arguments. Name is the argument name and Value is the corresponding value. Name must appear inside single quotes (' '). We can specify several name and value pair arguments in any order as Name1,Value1,...,NameN,ValueN.

Example: 'Color','yellow' specifies yellow for the shape color.

a.'LineWidth' — Shape border line width
1 (default) | positive scalar integer

Shape border line width, specified in pixels, as a positive scalar integer. This property only applies to the 'Rectangle', 'Line', 'Polygon', or 'Circle' shapes.

Data Types: uint8 | uint16 | int16 | double | single

b. 'Color' — Shape color
'yellow' (default) | character vector | cell array of character vectors | [R G B] vector | *M*-by-3 matrix

Shape color, specified as the comma-separated pair consisting of 'Color' and either a character vector, cell array of character vector, or matrix. We can specify a different color for each shape, or one color for all shapes.

To specify a color for each shape, set Color to a cell array of color character vectors or an *M*-by-3 matrix of *M* number of RGB (red, green, and blue) color values.

To specify one color for all shapes, set Color to either a color character vector or an [R G B] vector. The [R G B] vector contains the red, green, and blue values.

Supported colors: 'blue', 'green', 'red', 'cyan', 'magenta', 'black','black', and 'white'.

Data Types: cell | char | uint8 | uint16 | int16 | double | single

c. 'Opacity' — Opacity of filled shape
0.6 (default) | range of [0 1]

Opacity of filled shape, specified as the comma-separated pair consisting of 'Opacity' and a scalar value in the range [0 1]. The Opacity property applies for the FilledRectangle, FilledPolygon, and FilledCircle shapes.

Data Types: double | single | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64

d. 'SmoothEdges' — Smooth shape edges
true (default) | false

Smooth shape edges, specified as the comma-separated pair consisting of 'SmoothEdges' and a logical value of true or false. A true value enables an anti-aliasing filter to smooth shape edges. This value applies only to nonrectangular shapes. Enabling anti-aliasing requires additional time to draw the shapes.

Data Types: logical

IV. Output Arguments

RGB — Output image
M-by-*N*-by-3 truecolor
Output image, returned as a truecolor image.

CHAPTER 3: IMPLEMENTATION

3.1 Application of Image Processing License Plate Detection [IMEPLEMENTATION]

In Bangladesh it is a very highly populated country. Traffic jam is a common phenomenon in our country. We have two kinds of vehicles. One is private cars and others is bus and big vehicles. We have tried here to detect both type of number plate.

Capture:

The image is captured with a high resolution camera. In our project we have used a 8MP camera which is a back camera of a mobile phone. In this system there are more good camera is like (IR). A better choice could be the Infrared camera which is specially made for this kind of purpose. It is very costly that is why we did not effort it.

Fig: a number late of a car

Pre process:

It is a process where the algorithm is applied to the image. This algorithm basically starts with the image enhancing. It is very important for the computer vision. There are two steps.

1. Resize the image- the image which is captured by the camera it might be small or large. It affects the system. To reduced the problem it has been settled to resize the image.

2. Color convert- the picture might have the multimedia color but for the process we need to convert it n three colors. Red, green, blue. It is called the RGB mood. For the next step we need to convert it in RGB mood.

After this process is done it send to the next step.

Fig: RGB mood number plate picture.

License Plate extractor:

It is the most critical process. It has two parts.

1.Haar like features- It mainly recognize objects from the imge. It is a very old process. It take a long time to finish the process. It also make the system slow. If our project was only to detect the license plate then we used it.

2. Edge detection process-

We have used this process in our license plate extraction. In this process it makes a binary image to extract the image. It is more easier to extract the image. After following all steps it started t make a binary image of the captured image.

-Try to found the four connected point.

-Try to match the width/height ratio with the four connected point.

-The plate region is extracted from the image.

-The extraction of the license plate is performed.

This approach is very quick and smooth. After this it send to the next part.

Character Segmentation:

The extraction is done and it has removed the unnecessary data from the license plate. After the segmentation it has only the license plate character. It also do the width/height ratio matching and the four connected points.

Chapter: Implementation

Assumptions:

1. Maximum expected distance can be 5 meters.

2. The picture has to take in day light.

3. Minimum camera angle: 90 degree.

4. Minimum camera resolution: 3 mega pixel.

We will implement it in practical life. But for that we need more recourses.

3.2 Application of Image Processing in Traffic Control System

[IMPLEMENTATION]

vision.ForegroundDetector

Here ForegroundDetector method takes 4 parameters. First one is to declare the model we have used.Number of Gaussian modes in the mixture model, specified as the comma-separated pair consisting of 'NumGaussians' and a positive integer. Typically this value is 3, 4 or 5. Set this value to 3 or greater to be able to model multiple background modes. Last two parameters are to specify the number of training frames to detect the foreground.

Syntax:

```
foregroundDetector = vision.ForegroundDetector('NumGaussians', 3, ...
    'NumTrainingFrames', 50);
```

vision.VideoFileReader

This function is used to declare a videoReader object to read frame by frame of a particular video.

Syntax:

- videoReader = vision.VideoFileReader('MVI_6762.MOV');

step

This step function reads one frame at a time from the whole video.

Syntax:

- frame = step(videoReader);

Here step function is used to get the only the foreground of an image.

Syntax:

- foreground = step(foregroundDetector, frame);

strel

Next commented region was used to omit the unnecessary regions of the image. Like parts of the image which are not part of the main road can be omitted through these commented lines.

This function is used to declare a structural object to do morphological operations. Here we have used a squared shaped structural object with a dimension of 5*5.

Syntax:

- se = strel('square', 5);

imopen

This function does the morphological operation 'opening'. This is a procedure of doing erosion followed by one dilation. This function is used to remove noises from an image. It also disconnects objects which are connected through few number of pixels. So that unique objects are detected separately.

Syntax:

- filteredForeground = imopen(foreground, se);

vision.BlobAnalysis

This function is used to count the number of objects of an image. Here we are detecting objects with minimal area of 3000 pixels.

Syntax:

- blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, 'AreaOutputPort', true, 'CentroidOutputPort', true, 'MinimumBlobArea', 3000);

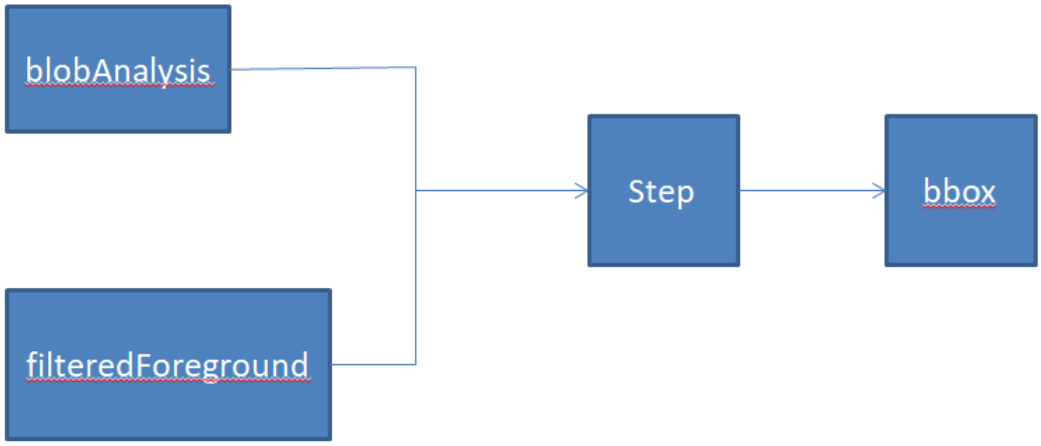
step

This step function is used to get the objects in the foreground image. The noise free foreground image we get from filterforeground and the area we declared in blobanalysis we take that as input and pass through step response. So through step response we detect the object in the foreground image of that declared area and store the objects in bbox.

Syntax

'bbox = step(blobAnalysis, filteredForeground);'

Block Diagram



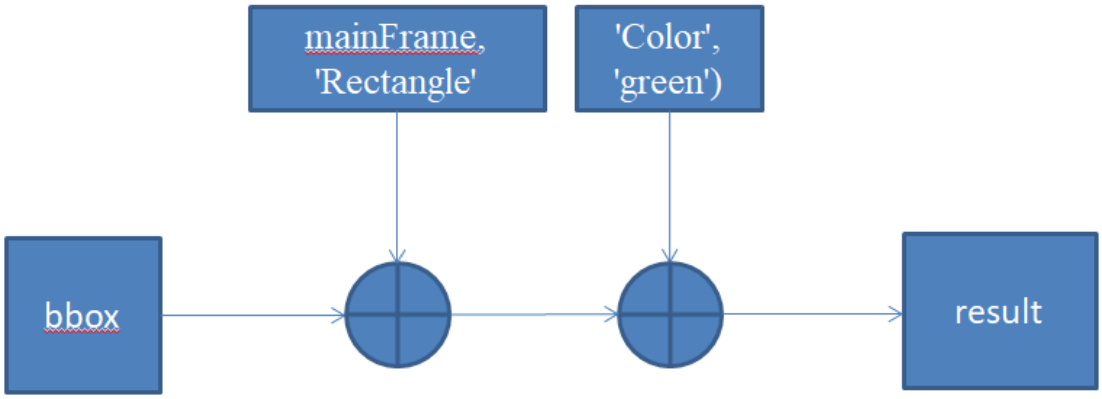
Insertshape

This functions is used to insert or mark the detected objects with a green colored rectangle box. The detected object in the foreground image of that specific declared area we stored in bbox, through this function we mark those objects. We mark those objects through a rectangle green colored box.

Syntax

'result = insertShape(mainFrame, 'Rectangle', bbox, 'Color', 'green');'

Block Diagram



Result

Here we can see that the objects are marked with a green colored rectangle box.

Loop

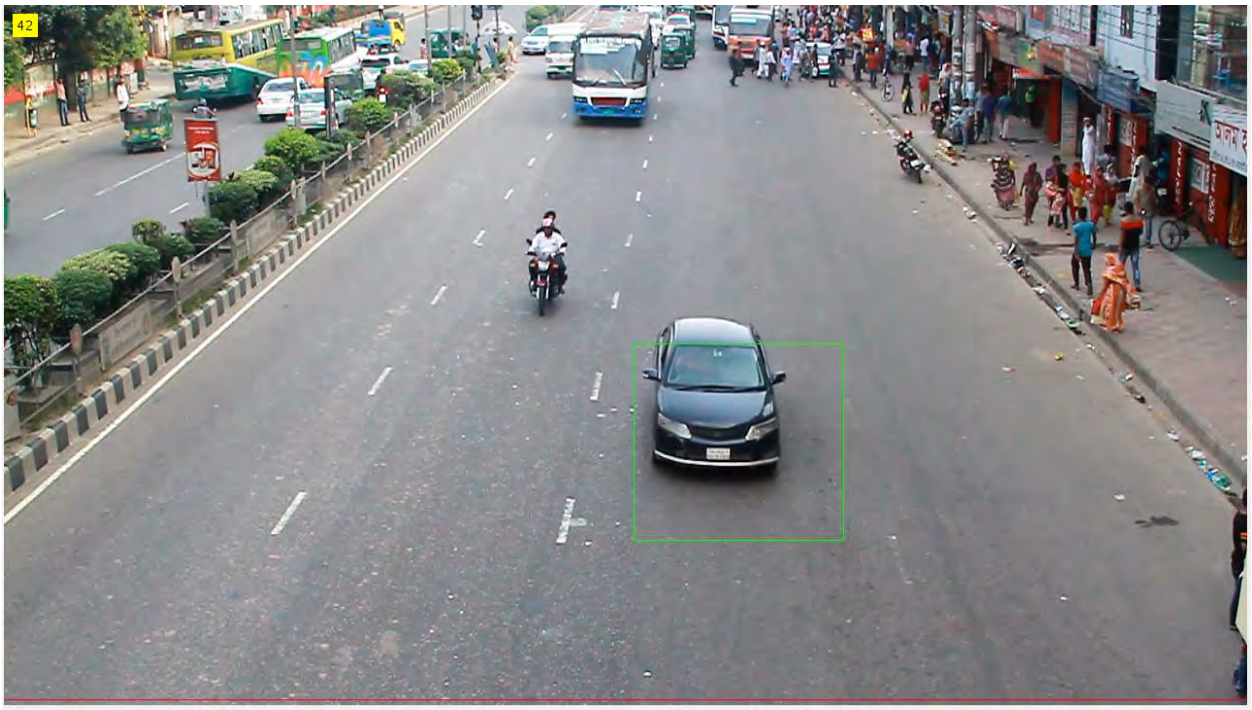
I have declared a small region to count the number of cars of an image. We are tracking the centroid of a detected object. Whenever the detected centroid is crossing that particular region

we are counting one vehicle. This region must be very small so that one vehicle is not being counted twice.

Syntax

```
‘for i=1:r
    if (centroids(i,2) > 676 && centroids(i,2) < 680 )
        numCars = numCars + 1;
    end
end’
```

Result



Here at the bottom of the picture we can visualize a red colored thin line. If a objects centroid passes through it, it will count one and so on. So thus we can get the actual number of vehicles passing through the line. At the left top of the image the total number of cars were displayed.

4.0 LIMITATIONS

4.1 Limitations

In conducting our project we had to face many difficulties. Some of which we were able to overcome or find an alternative to reach the completion of our project. In our license plate detection project, the maximum expected distance between number plate picture and camera was 1 feet. The camera could not detect any license plate situated more distance that this. The image we were detection must be in maximum brightness. Low light image was undetectable by the camera. For overcoming these lacking high resolution camera is needed, which was also a drawback. Our system is not automated. For more advanced future work and to use this system commercially we have to make the system automated so that everyone can use it easily. We did not get enough time to complete this though project. If we could get more time, we could have overcome many more drawbacks of our project. In the traffic monitoring project the most common problem we faced during detecting vehicles was the vehicles on the road was not maintaining lines. So it was difficult for us to shoot the video properly and also detecting those objects. Another major problem was people walking on the road and maintaining no rules. So when we were detecting vehicles at the limited distance some motorbikes and people were almost same size, so people passing through red line was also counting as object. This is a huge drawback. Different size of the vehicles was also a major problem. We did not had any data library to store the data's we got from the video. If we could have stored then we could easily detect stealing cars. Our error percentage was 26%. Total number of car passed through the red line was 57 which we manually counted. Our system detected 42. So the error percentage is 26%.

5.0 CONCLUSION & FUTURE WOKRS

5.1 Conclusions

The study showed that image processing is a better technique to control the state change of the traffic light and license plate deteection. It shows that it can reduce the traffic congestion, avoids the time being wasted by a green light on an empty road and also acts as security system. It is also more consistent in detecting vehicle presence because it uses actual traffic images. It visualizes the reality so it functions much better than those systems that rely on the detection of the vehicles' metal content. Overall, the system is good but it still needs improvement to achieve a hundred percent accuracy.

5.2 Future works

Our future goal is to contribute to our ever developing society by providing an advanced model of traffic monitoring system and license plate detection. For that we need more resources such as high resolution camera, advanced high definitions computers and more. We want our project to be used by government, so that we can have a protected and secured society. In future we are planning to make an app, so that people can easily find out from home that where is more traffic jam and which road to take. Recording license plate and detection just putting a number will reduce the job of police and they can easily catch thieves. They can also track terrorists and find out where they came from by detecting those vehicles number. We plan and expect to make further contributions to our research in future.

6.0 REFERNCE

- http://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf
- http://docs-europe.electrocomponents.com/webdocs/127d/0900766b8127db0a.pdf
- L.F. Shampine and P. Gahinet, "Delay-differential-algebraic equations in control theory," Applied Numerical Mathematics, Vol. 56, Issues 3–4, pp. 574–588
- https://www.mathworks.com/help/control/ref/step.html
- https://www.mathworks.com/help/vision/ref/insertshape.html
- https://www.researchgate.net/publication/305709395_Tracking_Multiple_Moving_Objects_Using_Gaussian_Mixture_Model
- https://www.mathworks.com/help/vision/ref/vision.foregrounddetector-class.html
- https://www.mathworks.com/help/vision/ref/vision.foregrounddetector.step.html
- https://www.mathworks.com/help/vision/ref/vision.videofilereader-class.html
- https://www.mathworks.com/help/images/structuring-elements.html
- https://www.mathworks.com/help/images/ref/strel-class.html
- https://www.mathworks.com/help/images/ref/imopen.html?s_tid=doc_ta
- https://www.google.com/search?q=alpr+block+diagram&client=firefox-b&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiAypmrz97VAhWCro8KHbJrB9MQ_AUICigB&biw=1138&bih=549#imgrc=_