

TRAFFIC SURVEILLANCE AND ESTIMATION USING IMAGE PROCESSING

Submitted in partial fulfillment of the Degree of
Bachelor of Technology



May – 2015

Enrollment. No. - 111008, 111029, 111093
Name of Student - Ankit Mamgain, Saurabh Srivastava, Mehak Goyal
Name of supervisor(s) - Ms. Pragya Gupta

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

TABLE OF CONTENTS

CHAPTER NO.	TOPICS	PAGE NO.
	Certificate	3
	Abstract	4
	Acknowledgement	5
Chapter 1	Introduction	6
Chapter 2	Image processing	8
2.1	Morphological Image processing	9
2.2	Image data basics	10
2.3	Dilation	13
2.4	Erosion	14
2.5	Opening	15
2.6	Closing	15
Chapter 3	Thresholding	16
3.1	Categories and Preliminaries	17
Chapter 4	Algorithm	24
Chapter 5	Results	30
5.1	Conclusion	35
Chapter 6	Code	36
	Appendix	42
	References	44
	Image References	45

CERTIFICATE

This is to certify that project report entitled “**TRAFFIC SURVEILLANCE AND ESTIMATION USING IMAGE PROCESSING**”, submitted by Ankit Mamgain (111008), Saurabh Srivastava (111029) and Mehak Goyal (111093) in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

DATE: 25/5/2018



Supervisor's Name: Ms Pragya Gupta

Designation: Assistant Professor (Grade-II)

ABSTRACT

Due to the increase in the number of vehicles day by day, traffic congestions and traffic jams are very common. One method to overcome the traffic problem is to develop a method to estimate the traffic density which is based on using real time image processing techniques. The theme is to control the traffic by determining the traffic density on each side of the road. This project presents the algorithm to determine the number of vehicles on the road and to determine the number on the number plate of the vehicle. The density counting algorithm works by comparing the real time frame of live video by the reference image and by searching vehicles only in the region of interest (i.e., road area). The computed vehicle density can be compared with other direction of the traffic in order to control the traffic signal smartly.

Name: Ankit Mamgain

Signature of Student

Name: Saurabh Srivastava

Signature of Student

Name: Mehak Goyal

Signature of Student

Signature of Supervisor

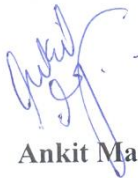
(Ms. Pragya Gupta)

Date:

ACKNOWLEDGEMENT

We are very grateful and highly acknowledge the continuous encouragement, invaluable supervision, timely suggestions and inspired guidance offered by our project supervisor **Ms. Pragya Gupta**, Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Waknaghat in bringing this project to a successful completion.

We are grateful to **Prof. T.S Lamba**, Dean, Academics & Research, Jaypee University of Information Technology and to **Prof. Sunil Bhooshan**, Head, Department of Electronics and Communication Engineering, Jaypee University of Information Technology for their continued support and encouragement. We offer our sincere appreciation for the continued support and encouragement. The completion of this project could not have been accomplished without the support of our teachers and friends who have always extended all sorts of help whenever needed.



Ankit Mamgain



Saurabh Srivastava



Mehak Goyal

CHAPTER 1

INTRODUCTION

In recent years, traffic congestion has become a significant problem. The simplest solution is to lay more lanes to reduce traffic density, but adding more lanes is not a feasible solution on account of time, cost and efficient utilization of the infrastructure. Detectors buried in the road, infra-red and radar sensors on the side provide limited traffic information and require separate systems for traffic counting and for traffic surveillance. Inductive loop detectors do provide a cost-effective solution, however they are subject to a high failure rate when installed in poor road surfaces, decrease pavement life and obstruct traffic during maintenance and repair. Infrared sensors are affected to a greater degree by fog than video cameras and cannot be used for effective surveillance. It is important to know the traffic density of the roads real time especially in mega cities for signal control and effective traffic management. Time estimation of reaching from one location to another and recommendation of different route alternatives using real time traffic density information are very valuable for mega city residents.

Detection of moving objects including vehicle, human, etc. in video can be achieved in three main approaches: Temporal difference, optical flow, and background subtraction. In temporal difference, the image differences of two consecutive image frames are obtained. However, this approach has some limitations such as visual homogeneity requirement and its effectiveness depends on the speeds of moving objects. Optical flow method was developed to obtain effective background modification, which bases on the detection of intensity changes. However, illumination change due to weather or sunlight reflections decreases its effectiveness. It is also computationally inefficient. The third method, background subtraction, is the mostly seen method in the literature for effective motion tracking and moving object identification. In background subtraction, background can be static, in which a fixed background is obtained beforehand and used in the entire process; or dynamic, in which background is dynamically updated with changing external effects like weather. Static background may not be effective in most applications; many methods include dynamic background subtraction. However, in

this project we are using the background as static as the camera is fixed at one position and its direction cannot be changed with changing weather and other environmental factors.

Image-based systems offer many advantages compared to traditional techniques. They provide more traffic information, combine both surveillance and traffic control technologies, are easily installed, and are scalable with progress in image processing techniques. The use of this technology is valuable for the analysis and performance improvement of road traffic.

There are lots of techniques proposed to design an intelligent traffic system but the best technique is based on the measurement of the traffic density by correlating the live traffic image with a reference image. The higher the difference is, higher traffic density is detected. The technique proposed in our project is based on computing the traffic load by comparing two images, the reference image and the live traffic image. This technique is proposed to estimate the traffic density by using image processing, in which we first select the reference image which is the image with no vehicles or less vehicles and every time matching real time images with that reference image. The calculation of number of vehicles is done by the method of object detection in the live time image. The result will be estimated by the difference of object in the live time image with the reference image. A GUI is developed which ultimately shows the number of vehicles present in the live image by drawing a square around them. As during the live application of this project, processing time also matters we are also generating a processing time result which shows us how much time our system will take to run the program and estimate the result. Obviously, more the system memory fast will be the processing time.

CHAPTER 2

IMAGE PROCESSING

In imaging science, image processing is any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it.

Image processing usually refers to digital image processing, but optical and analog image processing also are possible. The acquisition of images (producing the input image in the first place) is referred to as imaging.

Closely related to image processing are computer graphics and computer vision. In computer graphics, images are manually made from physical models of objects, environments, and lighting, instead of being acquired (via imaging devices such as cameras) from natural scenes, as in most animated movies. Computer vision, on the other hand, is often considered high-level image processing out of which a machine/computer/software intends to decipher the physical contents of an image or a sequence of images (e.g., videos or 3D full-body magnetic resonance scans). In modern sciences and technologies, images also gain much broader scopes due to the ever growing importance of scientific visualization (of often large-scale complex scientific/experimental data). Examples include microarray data in genetic research, or real-time multi-asset portfolio trading in finance. Digital image processing involves the manipulation and interpretation of digital images with the aid of a computer and it is an extremely broad subject and it often involves procedures, which can be mathematically complex.

The central idea behind is quite simple. The digital image is fed in to the computer one pixel at a time. The computer is programmed to insert the data in to an equation or series of equations, and then store the results that may display or further processed. Digital image processing used to solve a variety of

problems. Although often unrelated, these problems commonly require methods capable of enhancing pictorial information for human interpretation and analysis.

2.1 MORPHOLOGICAL IMAGE PROCESSING

Morphological Image Processing is an important tool in the Digital Image processing, since that science can rigorously quantify many aspects of the geometrical structure of the way that agrees with the human intuition and perception. Morphologic image processing technology is based on geometry. It emphasizes on studying geometry structure of image. We can find relationship between each part of image. When processing image with morphological theory. Accordingly we can comprehend the structural character of image in the morphological approach an image is analyzed in terms of some predetermined geometric shape known as structuring element.

Morphological processing is capable of removing noise and clutter as well as the ability to edit an image based on the size and shape of the objects of interest. Morphological Image Processing is used in the place of a Linear Image Processing, because it sometimes distorts the underlying geometric form of an image, but in Morphological image processing, the information of the image is not lost. In the Morphological Image Processing the original image can be reconstructed by using Dilation, Erosion, Opening and Closing operations for a finite no of times. The Morphological Image Processing is implemented and successfully tested in FORENSICS also.

The Morphological image processing is generally based on the analysis of a two valued image in terms of certain predetermined geometric shape known as structuring element. The term morphology refers to the branch of biology that deals with the form and structure of animals and plants. A very well suited approach for extracting significant features from images that are useful in the representation and description of region shapes is morphological (shape-based) processing. Morphological processing refers to certain operations where an object is Hit or Fit with structuring elements and thereby reduced to a more revealing shape. These structuring elements are shape primitives which are developed to represent some aspect of the information or the noise. By applying these structuring elements to the data using different algebraic combinations, one performs morphological transformations on the data.

2.2 IMAGE DATA BASICS

An image refers to a 2-D light intensity function, based on these 2-D arrays of numbers the images are categorized in to three forms,

- BINARY IMAGE
- GREY TONE IMAGE
- COLOR IMAGE

Binary Image: - The image data of Binary Image is Black and White. Each pixel is either '0' or '1'. A digital Image is called Binary Image if the grey levels range from 0 and 1.

Ex: A Binary Image shown below is,

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

FIG .1 4X5 Binary Image

READING THE TAG IMAGE FILES FORMAT:-

Image processing involves processing or altering an existing image in a desired manner. The first step is obtaining an image, while this may sound obvious; it is not a simple matter, since usable image data is not readily available.

The programmer needs a simple method of obtaining image data in a standard, usable format, called image file format. The image file format determines the image data storage and also gives additional

storage information with the pixel values. The image file consists of a Header Segment and a Data-Segment. The Header will contain, at the very least, the width and the height of the image. Since it is impossible to display or process any image without knowledge of its dimensions. The Headers of most file formats begin with a signature or magic number. A short sequence of bytes designed to identify the file as an image with the specific format.

FITTING AND HITTING:-

The Structuring Element is positioned at all positions or possible locations in the Binary Image and it is compared with the corresponding neighborhood of pixels. The morphological operation resembles a 'Binary' correction where the operation is logical than arithmetic in nature.

Ex.: Suppose we have two 3 * 3 structuring elements

$$\mathbf{S1} = \begin{matrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{matrix}$$

$$\mathbf{S2} = \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

In a given image A, B and C are the three positions where the S1 and S2 Structuring Elements should be positioned.

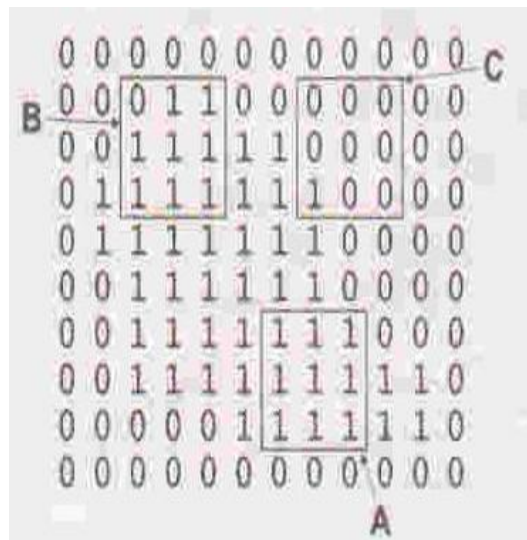


Fig.2 Binary Image used to test Fitting and Hitting of Structuring Elements S1 and S2.

FIT: - The structuring element is said to FIT the image if, for each of its pixels that is set to '1', the corresponding image pixel is also '1'. For the above example, Both S1 and S2 fit the image at 'A' (Remember that structuring element pixels set to '0' are ignored when testing for a fit). S2 fits the image at 'B' and neither S1 nor S2 fits at 'C'.

HIT: - A structuring element is said to HIT and Image if, for any of its pixels that is set to '1', The corresponding Image pixel is also '1'. (Here also we ignore Image pixels for which the corresponding structuring element pixel is '0'.) For the above example, S1 and S2 HIT the Image in neighborhood 'A'. The same holds true at 'B'. But at neighborhood 'C', only S1 HITS the Image. In this concept HITS corresponds to Union and where as the FITS corresponds to Intersection. Furthermore it is possible to replace the set operation Intersection and Union by the Boolean operators 'AND' and 'OR'.

2.3 DILATION

Dilation causes objects to dilate or grow in size. The amount and the way that they grow depend upon the choice of the structuring element. Dilation makes an object larger by adding pixels around its edges. The Dilation of an Image 'A' by a structuring element 'B' is written as $A \oplus B$. To compute the Dilation, we position 'B' such that its origin is at pixel co-ordinates (x, y) and apply the rule.

$$g(x, y) = \begin{cases} 1 & \text{if 'B' hits 'A'} \\ 0 & \text{Otherwise} \end{cases}$$

Repeat for all pixel co-ordinates. Dilation creates new image showing all the location of a structuring element origin at which that structuring element HITS the Input Image. In this it adds a layer of pixel to an object, there by enlarging it. Pixels are added to both the inner and outer boundaries of regions, so Dilation will shrink the holes enclosed by a single region and make the gaps between different regions smaller. Dilation will also tend to fill in any small intrusions into a region's boundaries.

The results of Dilation are influenced not just by the size of the structuring element but by its shape also.

Dilation is a Morphological operation; it can be performed on both Binary and Grey Tone Images. It helps in extracting the outer boundaries of the given images.

For Binary Image:-

Dilation operation is defined as follows,

$$D(A, B) = A \oplus B$$

Where,

A is the image

B is the structuring element of the order $3 * 3$.

Many structuring elements are requested for dilating the entire image.

2.4 EROSION

Erosion causes objects to shrink. The amount of the way that they shrink depends upon the choice of the structuring element. Erosion makes an object smaller by removing or eroding away the pixels on its edges.

The Erosion of an image 'A' by a structuring element 'B' is denoted as $A \ominus B$. To compute the Erosion, we position 'B' such that its origin is at image pixel co-ordinate (x, y) and apply the rule.

$$g(x, y) = \begin{cases} 1 & \text{if 'B' Fits 'A',} \\ 0 & \text{otherwise} \end{cases}$$

Repeat for all x and y or pixel co-ordinates. Erosion creates new image that marks all the locations of a Structuring elements origin at which that Structuring Element fits the input image. The Erosion operation seems to strip away a layer of pixels from an object, shrinking it in the process. Pixels are eroded from both the inner and outer boundaries of regions. So, Erosion will enlarge the holes enclosed by a single region as well as making the gap between different regions larger. Erosion will also tend to eliminate small extrusions on regions boundaries.

The result of erosion depends on Structuring element size with larger Structuring elements having a more pronounced effect & the result of Erosion with a large Structuring element is similar to the result obtained by iterated Erosion using a smaller structuring element of the same shape.

Erosion is the Morphological operation, it can be performed on Binary and Grey images. It helps in extracting the inner boundaries of a given image.

For Binary Images:-

Erosion operation is defined as follows,

$$E(A, B) = A \ominus B$$

Where,

A is the image

B is the structuring element of the order $3 * 3$.

Many structuring elements are required for eroding the entire image.

2.5 OPENING

It is a powerful operator, obtained by combining Erosion and Dilation. “Opening separates the Objects”. As we know, Dilation expands an image and Erosion shrinks it. Opening generally smoothes the contour of an image, breaks narrow Isthmuses and eliminates thin Protrusions.

The Opening of an image ‘A’ by a structuring element ‘B’ is denoted as $A \circ B$ and is defined as an Erosion followed by a Dilation, and is written as,

$$A \circ B = (A \ominus B) \oplus B$$

Opening operation is obtained by doing Dilation on Eroded Image. It is to smoothen the curves of the image. Opening spaces objects that are too close together, detaches objects that are touching and should not be, and enlarges holes inside objects.

Opening involves one or more Erosions followed by one dilation.

2.6 CLOSING

It is a powerful operator, obtained by combining Erosion and Dilation. “Closing, join the Objects”. Closing also tends to smooth sections of contours but, as opposed to Opening, it generally fuses narrow breaks and long thin Gulf’s, eliminates small holes and fills gaps in the contour.

The Closing of an image ‘A’ by a structuring element ‘B’ is denoted as $A \bullet B$ and defined as a Dilation followed by an Erosion; and is written as,

$$A \bullet B = (A \oplus B) \ominus B$$

Closing is obtained by doing Erosion on Dilated image. Closing joins broken objects and fills in unwanted holes in objects.

Closing involves one or more Dilations followed by one Erosion.

CHAPTER 3

THRESHOLDING

Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images. The simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity $I_{i,j}$ is less than some fixed constant T (that is, $I_{i,j} < T$), or a white pixel if the image intensity is greater than that constant. In the example image on the right, this results in the dark tree becoming completely black, and the white snow becoming complete white.

In many applications of image processing, the gray levels of pixels belonging to the object are quite different from the gray levels of the pixels belonging to the background. Thresholding becomes then a simple but effective tool to separate objects from the background. Examples of thresholding applications are document image analysis where the goal is to extract printed character logos, graphical content, musical scores, map processing where lines, legends, characters are to be found, scene processing where a target is to detected , quality inspection of materials. Other applications include cell images and knowledge representation, segmentation of various image modalities for non-destructive testing (NDT) applications, such as ultrasonic images in, eddy current images, thermal images; X-ray computed tomography (CAT), laser scanning co focal microscopy, extraction of edge field, image segmentation in general, spatial-temporal segmentation of video images etc.

The output of the thresholding operation is a binary image whose gray level of 0 (black) will indicate a pixel belonging to a print, legend, drawing, or target and a gray level of 1 (white) will indicate the background.

The main difficulties associated with thresholding such as in documents or NDT applications occur when the associated noise process is non-stationary, correlated and non-Gaussian. Other factors complicating thresholding operation are ambient illumination, variance of gray levels within the object and the background, inadequate contrast, object shape and size non-commensurate with the scene. Finally the lack of objective measures to assess the performance of thresholding algorithms is another handicap. In fact most authors limit themselves to the visual inspection of a few test cases.

A document image analysis and recognition system includes several image processing techniques, beginning with digitization of the document and ending with character recognition and natural language processing. Thresholding is one of the first low-level image processing techniques used, before

document analysis step, for obtaining a binary image from its gray scale one. The thresholding step can be quite critical in that it will affect the performance of successive steps such as segmentation of the document into text objects, and the correctness of the OCR (optical character recognition). Improper thresholding causes blotches, streaks, erasures on the document confounding segmentation and recognition tasks. The merges, fractures and other deformations in the character shapes as a consequence of incorrect thresholding are known to be the main reasons of OCR performance deterioration. In turn thresholding algorithms depend on a multitude of factors such as the gray level distribution of the document, local shading effects, the presence of denser, non-text components such as photographs, the quality of the paper etc. In this study we develop taxonomy of thresholding algorithms based on the type of information used. We distinguish six categories, namely, thresholding algorithm based on the exploitation of

- 1) **Histogram entropy information,**
- 2) **Histogram shape information,**
- 3) **Image attributes information such as contours,**
- 4) **Clustering of gray-level information,**
- 5) **Locally adaptive characteristics,**
- 6) **Spatial information.**

3.1 CATEGORIES AND PRELIMINARIES

We categorize the thresholding methods in six groups according to the information they are exploiting. These categories are:

1. **Histogram shape-based methods** where the peaks, valleys and curvatures of the smoothed histogram are analyzed.
2. **Clustering-based methods** where the gray level samples are clustered in two parts as background and foreground (object) or alternately are modeled as two Gaussian distributions.
3. **Entropy-based methods** result in algorithms, for example, that use the entropy foreground-background regions, the cross-entropy between the original and binarized image etc.
4. **Object attribute-based methods** search a measure of similarity between the gray-level and binarized images, such as fuzzy similarity, shape, edges, number of objects etc.

5. **The spatial methods** use the probability mass function models taking into account correlation between pixels on a global scale.

6. **Local methods** do not determine a single value of threshold but adapt the threshold value depending upon the local image characteristics.

In the sequel we use the following notation. The histogram and the probability mass function (pmf) of the image are indicated, respectively, by $h(g)$ and by $p(g)$, $g = 0..G$, where G is the maximum luminance value in the image, typically 255 if 8-bit quantization is assumed. If the gray value range is not explicitly indicated as $[g_{\min}, g_{\max}]$ it will be assumed to extend from 0 to G . The cumulative probability function is defined as $P(g) = \sum_{i=0}^g p(i)$. It is assumed that the pmf is estimated from the

histogram of the image by normalizing to the number of samples at every gray level. In the context of document processing, the foreground (object) is the set of pixels with luminance values less than T , while the background pixels have luminance value above this threshold.

The foreground (object) and background pmf's will be expressed as $p_f(g)$, $0 \leq g \leq T$, and $p_b(g)$, $T+1 \leq g \leq G$, respectively, where T is the threshold value. The foreground and background area probabilities are calculated as:

$$P_f(T) = P_f = \sum_{g=0}^T p(g) \qquad P_b(T) = P_b = \sum_{g=T+1}^G p(g) \qquad (1)$$

The Shannon entropy parametrically dependent upon the threshold value T for the foreground and background is formulated as:

$$H_f(T) = - \sum_{g=0}^T p_f(g) \log p_f(g) \qquad H_b(T) = - \sum_{g=T+1}^G p_b(g) \log p_b(g) \qquad (2)$$

The sum of these two is expressed as $H(T) = H_f(T) + H_b(T)$. When the entropy is calculated over the input image distribution $p(g)$ (and not over the class distributions), then obviously it does not depend upon the threshold T and hence is expressed simply as H . For various other definitions of the entropy in the context of thresholding, with some abuse of notation, we will use the same symbols of $H_f(T)$ and $H_b(T)$.

The fuzzy measures attributed to the background and foreground events, that is the degree to which the gray level; g , belongs to the background and object, respectively, are symbolized by $\mu_f(g)$ and $\mu_b(g)$.

The mean and variance of the foreground and background as functions of the thresholding level T can be similarly denoted as:

$$m_f(T) = \sum_{g=0}^T g p(g) \quad \sigma_f^2(T) = \sum_{g=0}^T (g - m_f(T))^2 p(g) \quad (3)$$

$$m_b(T) = \sum_{g=T+1}^G g p(g) \quad \sigma_b^2(T) = \sum_{g=T+1}^G (g - m_b(T))^2 p(g) \quad (4)$$

HISTOGRAM SHAPE-BASED THRESHOLDING METHODS

This group of thresholding methods is based on the form and shape properties of image histograms. Rosenfeld and de la and Lee et al used histogram concavity analysis to derive the optimal threshold value for a given image. In their paper, a convex 5 hull of the image histogram is calculated and the deepest concavity points are selected as candidates to be the threshold value. In order to make computations faster, ad-hoc hardware units are designed that are able to compute a value for a standard-sized frame in about 10 milliseconds. In Seazan, the histogram function is convolved with a smoothing kernel. The gray levels where the peaks start, end and attain maxima are estimated. To add some data reduction, the algorithm sets gray-level thresholds between the peaks, and the gray levels at which the peaks attain a maximum are chosen as quantization levels. In Chang et al., a multi-modal histogram thresholding method is proposed based on a combination of regularization and statistical approaches. The original histogram is decomposed in several non-overlapping distributions by modeling it with a mixture of Gaussian density. Although the histogram is contaminated by contiguous distributions, experiments shown on the paper with simulated data demonstrate that this method outperforms similar ones at the task of finding the best threshold values and the parameters of predefined distributions. It is also a good option for real-world images as they generally do not come as a Gaussian mixture of densities. The only caveat in this paper is the heuristic nature of the smoothness factor, which would generate different thresholding values depending on the particular election. There is opportunity for future work regarding smoothness factors for different gray-level images. Ramesh et al were among the first who used a simple two-step approximation function to the normalized histogram. Thus, the sum of squares between the function and the histogram is minimized and the optimal threshold value can be obtained by performing an iterative search. Prewitt and Mendelsohn, in their analysis of cell images, proposed a method that iteratively smoothed the histogram using a running average of size 3, until there are two local maxima, j and k . The final threshold t is computed as the average, $(j + k)/2$.

CLUSTERING-BASED METHODS

These methods rely on generating clusters from gray-level information. Because the end result is a binary image, there are only two classes, or clusters. Each cluster corresponds to a lobe of the histogram. There are several clustering approaches which can be further subclassified as follows:

Iterative-based methods: Iterative schemes are based on the mixture of two Gaussian-based models.

The basic theory behind iterative thresholding can be summarized as follows:

1. Choose an initial threshold (T), either randomly or using another method.
2. Segment the image as background and foreground according to this initial threshold.
 - (a) $G1 = \{f(x, y): f(x, y) > T\}$
 - (b) $G2 = \{f(x, y): f(x, y) \leq T\}$
3. Compute the average of each set. Let m1 be the average of G1 and m2 be the average of G2.
4. A new threshold, T', is calculated from the average of m1 and m2.
5. Repeat step 2 until the difference between T and T' is small enough.

This method is known as Calvard et al., and has undergone several modifications and improvements from the research community. One of those modifications was developed by Yanni and E. In their work, the proposed iterative algorithm is initialized to the midpoint between the two peaks of the histogram, the highest gray level and the lowest one.

Clustering thresholding: In this category we can fit in the Otsu's method, one the most referenced thresholding methods in the literature. Otsu's method is based on selecting a threshold for separating the image into two classes so that the variance within each class is minimized. For obvious reasons, the distributions cannot be changed, but the selection of a threshold value modifies the spread of the two parts of the distribution. The goal is to select a threshold that minimizes the combined spread.

$$\sigma_{within}^2(T) = n_B(T)\sigma_B^2(T) + n_F(T)\sigma_F^2(T)$$

Where:

$$n_B(T) = \sum_{i=0}^{T-1} p(i)$$

$$n_F(T) = \sum_{i=T}^{N-1} p(i)$$

$$\sigma_B^2(T) = \text{The variance of background pixels}$$

$$\sigma_F^2(T) = \text{The variance of foreground pixels}$$

The above equations require the computation of within-class variance for each class and for each possible thresholding value, resulting in a very expensive computation that must be avoided.

Minimum error thresholding: Another way to minimize the error in classification is to suppose that each group or cluster is Gaussian-distributed with a mean and variance independent of the chosen threshold. Whereas the Otsu's method separates the image into two clusters according to the threshold and then tries to optimize some statistical measures, minimum error thresholding methods suppose there is a distribution and we have to estimate its parameters. If the two distributions are well separated (there could be a little overlap), we can reasonably assume that if we choose an arbitrary threshold, the mean and standard deviation of each group should approximate the mean and standard deviation of the underlying populations. The optimal threshold can be characterized as the one that causes the mixture of the two Gaussians to better approximate the real histogram. To reduce the solution space of this problem, a gradient descent method is used from an initial guess. Otsu's method can be a good technique for estimating that initial guess. Kittler and Illingworth formulated one of the principal minimum error methods. In recent papers, Cho et al. have suggested some improvements by detecting a bias that affects the threshold calculation.

Fuzzy clustering thresholding: Ramesh et al. approach the thresholding problem by assigning fuzzy clustering memberships to pixels in an image depending on the differences between the mean of the two classes. In their work two thresholding schemes are proposed. The first one is based on minimizing the variance of the approximated histogram. The second one is based on minimizing the sum of square errors. Experimental results show that the former scheme gives better results on average than the latter one, at the expense of a small additional computational cost.

ENTROPY-BASED METHODS

In this category we can enclose methods that use the entropy of the distribution of gray levels in the picture. The entropy is a concept that comes from the second law of Thermodynamics and measures spontaneous dispersal of energy. It was later introduced to communications theory by Shannon as a measure of the efficiency in data transmission over a noisy channel. High entropy is indicative of a great information transfer. The opposite consideration, preservation of information, can be achieved by minimizing cross-entropy between the input, gray level image and the output, thresholded image. Kapur et al. used Shannon's concept of entropy, from a different point of view. They considered the background and the foreground as two different image signals. Each of those classes has their entropy calculated and summed, so that when the sum is maximum the threshold is considered optimal.

OBJECT ATTRIBUTE-BASED METHODS

These methods extract a threshold value based on similarity between the original image and the binarized one using some attribute quality or similarity measure. Some of those measures are gray-level moments and fuzzy measures.

Moment preserving thresholding: In Tsai's work, the notion of similarity between source and target images is suggested by considering the source, gray-level image, and a blurred version of an ideal binary image. The thresholding is computed by matching the first three gray-level moments with the first three moments of the binary image. Further improvements by Cheng and Tsai used neural networks as an aid to the algorithm.

LOCALLY ADAPTIVE THRESHOLDING

This class of thresholding algorithms calculates a threshold value for each pixel, depending on some local parameters like range, variance or surface-fitting in the neighborhood. Among the first adaptive thresholding techniques was the algorithm developed by Nakagawa and Rosenfeld, who proposed a variation on the original method of variable thresholding by Chow and Kaneko. In their study, the image is divided into several windows, or subsets of the original image with the locality property. Those windows with bimodal histograms are selected, and a threshold is calculated. Threshold from different windows are interpolated to calculate a threshold for the whole image. The above method was successfully applied to TV images of machine components, with results substantially better than those that applied a fixed threshold to the whole image. A further extension allowed for trimodal histograms in the computation of the threshold, which yielded better results with the negative point of being more sensitive to shadows.

Local variance methods: Niblack developed an algorithm that adapts threshold selection to local mean $m(i, j)$ and standard deviation $\sigma(i, j)$ and a local window of size $b \times b$. This method proved to work well for optical character recognition (OCR) tasks. Further improvements introduced by Sauvola and Pietikäinen changed the impact of the standard deviation in the algorithm to better recognize letters in stained or documents with bad illumination.

Local contrast methods: White and Rohrer suggested comparing the gray level of a pixel with the average of gray levels of their neighbor pixels. For their experiments they used a window which approximates roughly to the size of a printed character. If a pixel is significantly darker than the average, it can be classified as a character pixel; otherwise, it is a background pixel. Venkateswarlu and Boyle are a great reference for performance comparison of local adaptive methods. Huang's method

(Huang and Wang) first smoothes the image by averaging the gray level of a pixel with their local neighbors, provided that the range (difference between maximum and minimum gray levels within the window) is below a given threshold T_1 . Next, an adaptive threshold is applied, which sets a pixel to the maximum value if it is greater than the local average, or if the local range is below a threshold T_2 .

Surface-fitting thresholding: In Yanowitz's method (Yanowitz and Bruckstein), gray-level information is combined with edge information to build a threshold surface. This method that exploits geometric features of blueprint images. It works under the assumption that the objects have a "c" shape, and the area is small. Multiple window sizes are proposed in the work, which effectively reduces computation time and helps distinguish thin lines from thick lines. Although the method is targeted at blueprint images, results indicate that it is also good for a wide range of images. The threshold surface is constructed by interpolation with potential surface functions and it is obtained iteratively using a discrete Laplacian on the surface. Other clever methods, specially used for badly illuminated images, include the one proposed by Parker. This method involves a first step where objects are located in the scene by using an intensity gradient. Next, levels that correspond to the objects in the various parts of the image are used as initial guesses for the threshold. Overall, Parker's method outperforms many classical adaptive threshold algorithms when applied on images produced with variable illumination.

Kriging method: Oh's method (Oh and Lindquist) is a two-pass algorithm that firstly uses a non-local thresholding algorithm such as Kapur et al. to classify the majority of the pixel population in one of two classes (object and background). Then, a variation of Kapur's method is applied, in which a lower threshold is established, below which gray levels are assigned to the first class (for example, the object class). Then, a second, larger, threshold value is found such that any pixel with a greater gray level is automatically assigned to the second class (for example, the background class). The remaining pixels whose gray level values lie between the two thresholds are left to the second pass. In the second pass, also known as the indicator kriging stage, pixels are assigned one of the two classes using local covariance of the class indicators and the constrained linear regression technique called kriging, within a region.

CHAPTER 4

ALGORITHM

Our project works on calculating the density of the traffic on a road. The work is divided into 4 parts. The first part is to process the image from fixed camera which will give us the live time image. A previous taken image of an empty road will be our reference image. The second part is to select the target area where the vehicles could be present by using image cropping technique. The third part is the object detection which is performed by enhancing features of the image. Finally, the last part is the density counting, where the numbers of vehicles are being counted. Next part will be extracting number plate from the image which will be explained later.

Following is the flow chart of the working of our project:

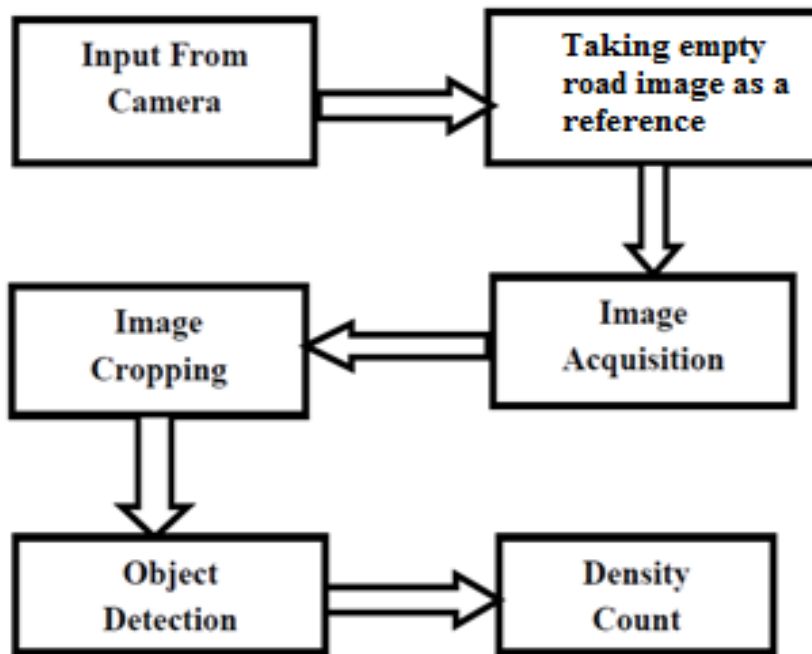


FIG .3 Flowchart

FOLLOWING ARE THE BRIEF INTRODUCTION OF THE STEPS INVOLVED IN THE PROCESS:

1. Processing of Image and Image Acquisition

The work starts with processing the live image using MATLAB software. The camera is stationary, which is mounted on the pole near the traffic signal. In our case, we are using a webcam which acts as the camera which will be mounted near the traffic signal. The next stage is to extract the frames continuously from the real time image coming from the stationary camera. This raw digital data is further processed by converting the images from RGB (Red-Green-Blue) to grayscale in order to further process the images. Initially the system captures the image of a vacant road when there is no vehicle present; this image is used as a reference image. Also, we have built a structure which we are using as a road. The camera will be mounted on the pole with a certain angle given to it so that there is only the dummy road in the frame of the camera. In real time, same technique will be followed.

The specifications of the camera we are using are:



FIG .4 Quantum QHM500-8LM(S) WEB CAMERA

Specifications:

1. Interpolated to 30 Mega Pixels
2. 10 Level Zoom on live Motion Picture
3. Special Visual Effects
4. True Motion Picture
5. Night Vision 8 Bright light switch ON through switch and potentiometer
6. Inbuilt Sensitive Microphone
7. Background Changeable of Live Motion Picture
8. UP TO 30% Better Exposure To Give better Picture Even in dark
9. Auto Exposure
10. Special Face Effects
11. USB 2.0

Resolution Hardware :	500K pixels
Image Quality:	RGB24 or I420
Exposure:	Auto or manual
	Angle of View 58 o
Interface:	USB2.0
Frame Rate:	30 fps (MAX)
Lens:	f=6.0 F=2.0
Focus Range	4cm to infinity

Basic requirements of the camera:

1. 1 GHz
2. 512 MB RAM or more
3. 200 MB hard drive space
4. Internet connection
5. USB 1.1 port (2.0 recommended)

Supports:

Windows Vista, Windows 7 (32-bit or 64-bit) or Windows 8

Dimensions:

Width x Height x Depth 13.5 x 19.5 x 4.5 cm

Weight 260/100 g

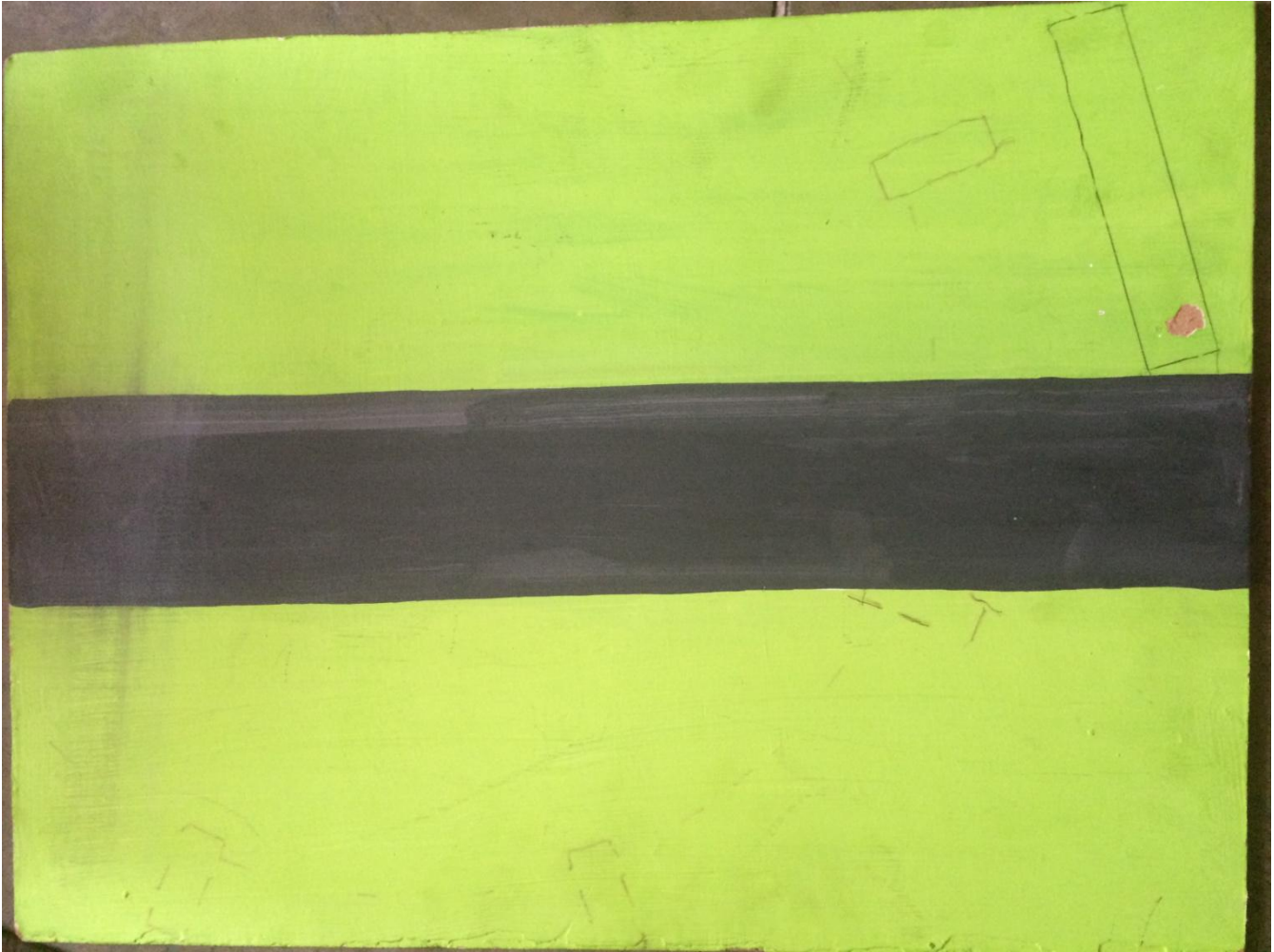


FIG .5 DUMMY ROAD

2. Image Cropping

The second step is to select the targeted area by designing image cropping algorithms in MATLAB. The purpose of cropping is to identify the road region where the vehicles are present and exclude the unnecessary background information. This unnecessary information is fixed in every image because the camera is stationary. To crop the required area, reference image has been used which has no road traffic. First, a binary image of having the same dimensions is created, as in the reference image, and then the road area has been shaded white, and the leftover region as black. Finally, the multiplication of the reference image with the cropping black and white image results in the final desired target area

3. Object Detection

The third step is the object or vehicle detection in order to identify and count the vehicles which are present in the targeted area. To perform the object detection, first the frame from the real

time image is taken. The next step is to convert both images; the reference image and the real time image into grayscale and then the absolute difference of two images will be determined. Since the dimensions of the road are fixed therefore the difference image only highlights the presence of vehicles in the desired target area. In order to improve the visibility of the vehicles, the difference image is converted to a binary image based on a threshold value. The resulting binary image where the presence of any object is more improved. In order to determine only vehicles in the desired area, multiplication of the cropped image with the enhanced version of the difference image is carried out. In the product image the unnecessary information is filtered out and it only highlights the presence of vehicles in the desired area.

4. Traffic Density

The next step is to calculate the traffic density in the desired target area. In order to determine the traffic density, the vehicles are marked first and then their numbers are counted. The algorithm search for a set of connecting pixels. In order to consider a connected region as a vehicle, a minimum threshold has been defined. However, it is possible that more than one region of a vehicle is detected using the above criteria. This problem could be overcome by finding the overlapping bounding boxes of the selected regions and thus smaller and highly overlapping regions are filtered out. The results where each detected vehicle is surrounded by a bounding box and the top-left region shows the number of vehicles detected on the road.

The next part of the project works on extracting the number plate from the image given as an input.

The algorithm is as follow:

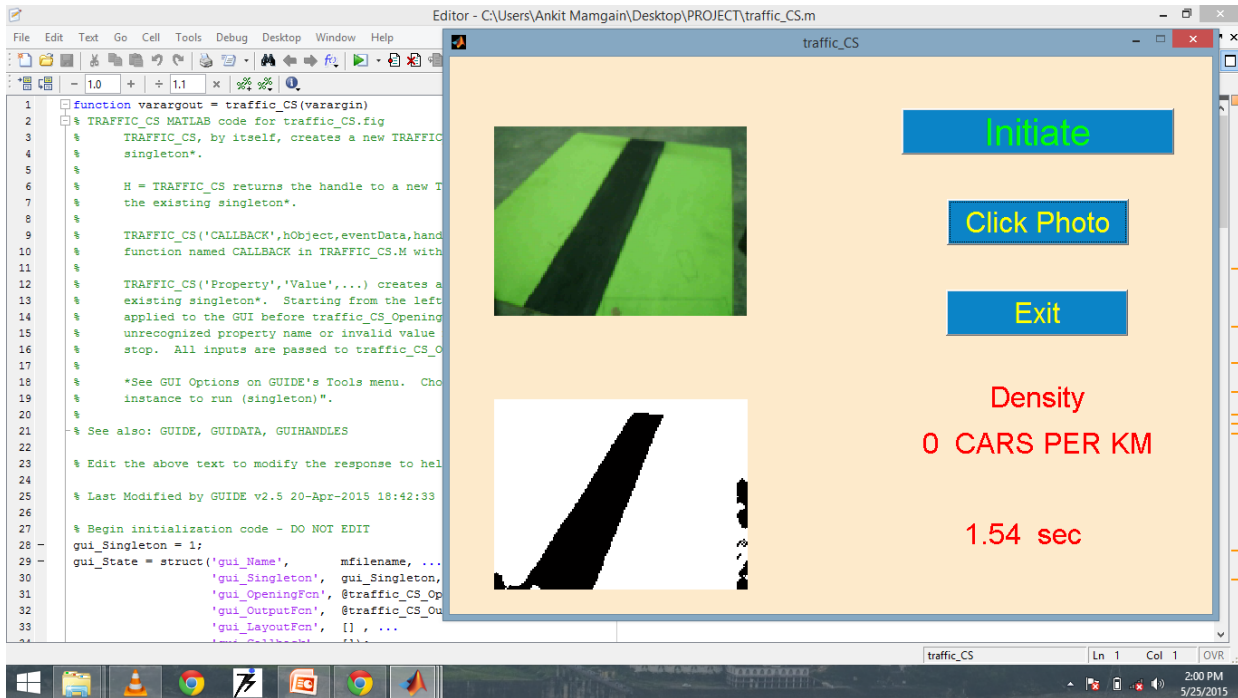
1. Reading the number plate image.
2. Resizing the image keeping aspect ratio same.
3. Converting the RGB (color) image to gray (intensity).
4. Median filtering to remove noise.
5. Structural element (disk of radius 1) for morphological processing.
6. Dilating the gray image with the structural element.
7. Eroding the gray image with structural element.
8. Morphological Gradient for edges enhancement.
9. Converting the class to double.
10. Convolution of the double image for brightening the edges.
11. Intensity scaling between the ranges 0 to 1.
12. Conversion of the class from double to binary.
13. Eliminating the possible horizontal lines from the output image of region grow that could be edges of license plate.
14. Filling all the regions of the image.
15. Thinning the image to ensure character isolation.
16. Selecting all the regions that are of pixel area more than 100.
17. Uncomment to make compitable with the previous versions of MATLAB® Two properties 'BoundingBox' and binary 'Image' corresponding to these Bounding boxes are acquired.

18. Selecting all the bounding boxes in matrix of order $\text{numberofboxes} \times 4$.
19. Calling of controlling function.
20. Function 'controlling' outputs the array of indices of boxes required for extraction of characters.
21. If successfully indices of desired boxes are achieved.
22. Cell array of 'Image' (one of the properties of regionprops)
23. Initializing the variable of number plate string.
24. Extracting the binary image corresponding to the indices in 'r'.
25. Reading the letter corresponding the binary image 'N'. Since it wouldn't be easy to distinguish between '0' and 'O' during the extraction of character in binary image. Using the characteristic of plates in India that starting three characters are alphabets, this code will easily decide whether it is '0' or 'O'. The condition for 'if' just needs to be changed if the code is to be implemented with some other cities plates. The condition should be changed accordingly.
26. Appending every subsequent character in noPlate variable.
27. This portion of code writes the number plate to the text file, if executed a notepad file with the name noPlate.txt will be open with the number plate written.

CHAPTER 5

RESULTS

1. Reference image with no traffic (FIG. 6)



2. Results obtained when there are vehicles present on the road. (FIG 7)



3. Road with one vehicle detected(FIG. 8)



4. Road with three vehicles detected (FIG.9)



5. Road with four vehicles detected. (FIG 10)



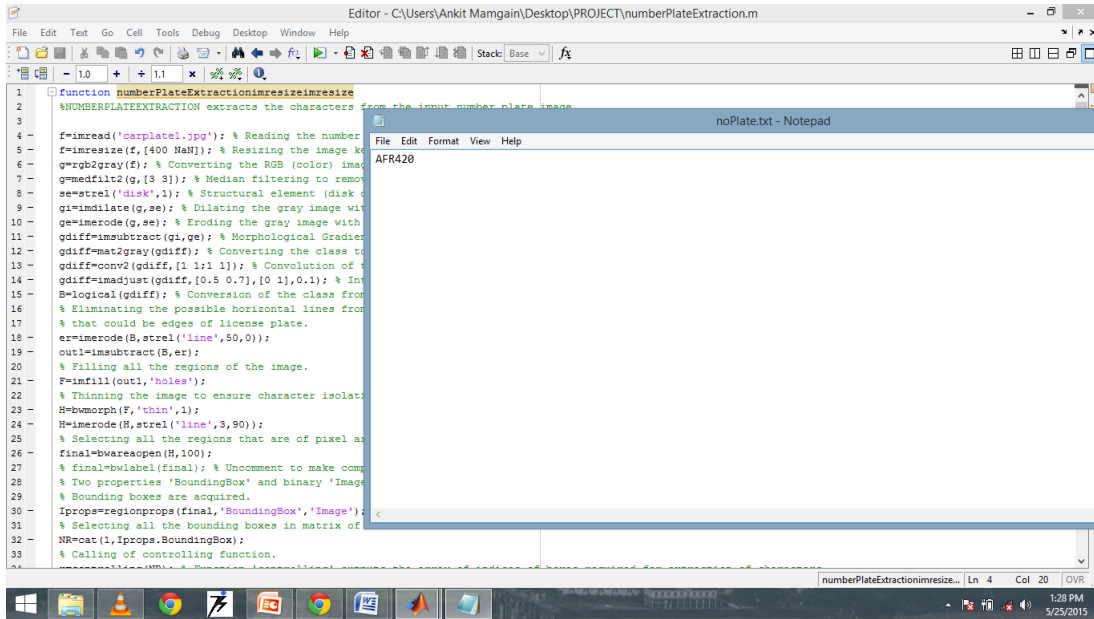
These were the results obtained after placing dummy vehicles on the road.

NUMBER PLATE EXTRACTION:

INPUT IMAGE FOR NUMBER PLATE EXTRACTION: (FIG 11)



OUTPUT OBTAINED: (FIG 12)

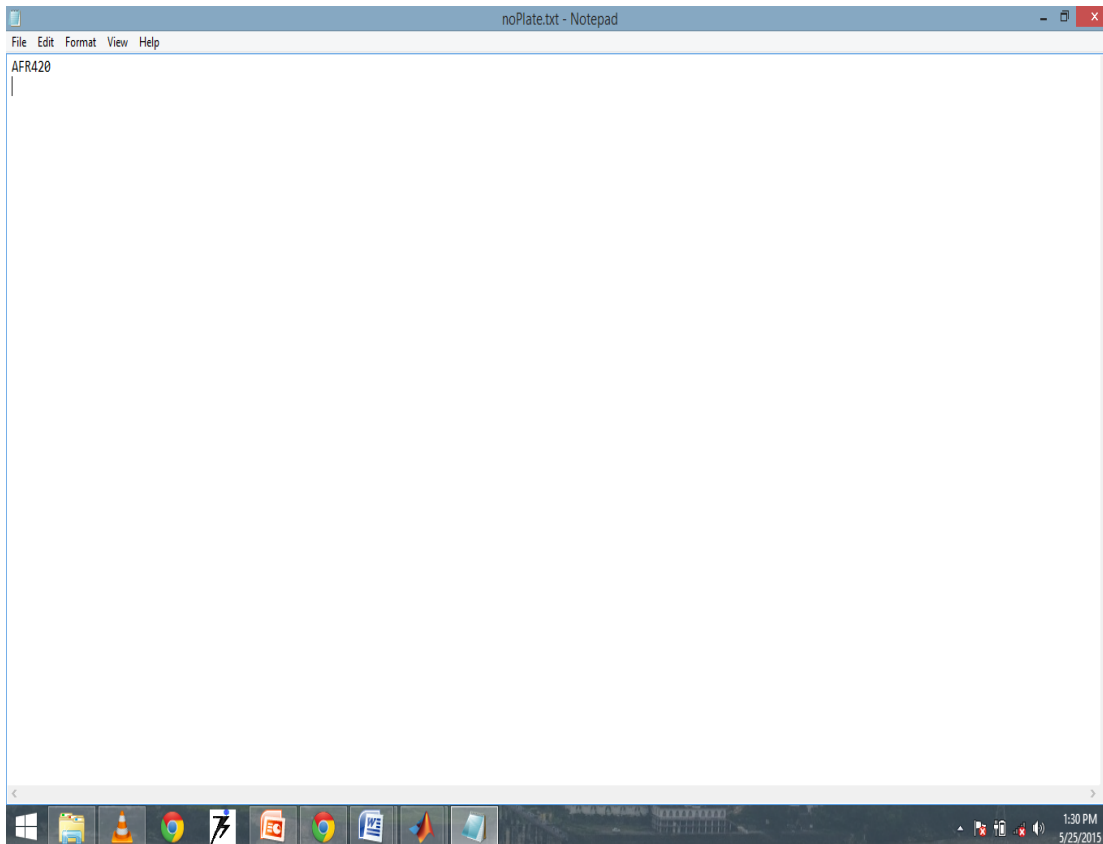


The screenshot shows a MATLAB Editor window with a function named `numberPlateExtractionImresizeImresize`. The function performs the following steps:

- Reads the input image `carplate1.jpg`.
- Resizes the image to `[400 NaN]`.
- Converts the image to grayscale (`rgb2gray`).
- Applies median filtering (`medfilt2`).
- Defines a structural element (`strel('disk',1)`).
- Dilates the image (`imdilate`).
- Erodes the image (`imerode`).
- Calculates the morphological gradient (`imsubtract`).
- Converts the gradient to grayscale (`mat2gray`).
- Convolution with `[1 1 1]`.
- Adjusts contrast (`imadjust`).
- Converts to binary (`im2bw`).
- Eliminates horizontal lines (`erode`).
- Fills holes (`imfill`).
- Thins the image (`bwmorph`).
- Erodes the image (`imerode`).
- Selects regions of pixel area `H` (`bwareaopen`).
- Uncomments the final image (`bwlabel`).
- Acquires bounding boxes (`regionprops`).
- Selects bounding boxes (`select`).
- Concatenates the bounding boxes (`cat`).

A Notepad window titled `noPlate.txt` is open, displaying the extracted license plate number `AFR420`.

FIG 12(a)



The screenshot shows a Notepad window titled `noPlate.txt` with the text `AFR420` entered.

Second input for vehicle's number plate detection: (FIG 13)



Output obtained: Fig 14

```
Editor - C:\Users\Ankit Mamgain\Desktop\PROJECT\numberPlateExtraction.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack Base fx
function numberPlateExtractionimresizeimresize
%NUMBERPLATEEXTRACTION extracts the characters from the input number plate image.
1
2
3
4 f=imread('carplate4.jpg'); % Reading the image
5 f=imresize(f,[400 NaN]); % Resizing the image
6 g=rgb2gray(f); % Converting the RGB (color) image to gray
7 g=medfilt2(g,[3 3]); % Median filtering to remove salt and pepper noise
8 se=strel('disk',1); % Structural element
9 gi=imdilate(g,se); % Dilating the gray image
10 ge=imerode(g,se); % Eroding the gray image
11 gdif=imsubtract(gi,ge); % Morphological gradient
12 gdif=mat2gray(gdif); % Converting the gradient to gray
13 gdif=conv2(gdif,[1 1 1]); % Convolution with a 1x3 kernel
14 gdif=imadjust(gdif,[0.5 0.7],[0 1],0.1); % Contrast stretching
15 B=logical(gdif); % Conversion of the gray image to binary
16 % Eliminating the possible horizontal lines
17 % that could be edges of license plate.
18 er=imerode(B,strel('line',50,0));
19 out1=imsubtract(B,er);
20 % Filling all the regions of the image.
21 F=imfill(out1,'holes');
22 % Thinning the image to ensure character
23 H=bwmorph(F,'thin',1);
24 H=imerode(H,strel('line',3,90));
25 % Selecting all the regions that are of pixels
26 final=bwareaopen(H,100);
27 % final=bwlabel(final); % Uncomment to make
28 % Two properties 'BoundingBox' and binary
29 % Bounding boxes are acquired.
30 Iprops=regionprops(final,'BoundingBox','Area');
31 % Selecting all the bounding boxes in matrix
32 NR=cat(1,Iprops.BoundingBox);
33 % Calling of controlling function.
34
```

```
noPlate.txt - Notepad
File Edit Format View Help
374HQR
```

numberPlateExtractionimresize... Ln 4 Col 20 OVR

2:21 PM 5/25/2015

5.1 CONCLUSION

This project discusses a method for estimating the traffic density on the lane by using image processing. The proposed system is very cost effective as it does not require installation of any additional devices, such as RFIDs. This work can be enhanced further by proposing a system which identifies the presence of emergency vehicles (like an ambulance or fire brigade) and by giving preference to those emergency vehicles. Secondly, it can be enhanced by using VANETs (Vehicular Ad-hoc Networks) as it provides road safety and intelligent transport system. If the number of cars exceeds a specific threshold, warning of heavy traffic will be shown automatically. The advantages of this new method include such benefits as:

- 1) Non-use of sensors
- 2) Low cost and easy setup and relatively good accuracy and speed.
- 3) There is no need to use aerial imagery.

Because this method has been implemented using Image Processing and Matlab software, production costs are low while achieving high speed and accuracy.

Further, the number plate extraction task can be linked with traffic estimation project while using high resolution camera. By using high resolution camera, we can easily extract the number plate by zooming in on the image captured. Same MATLAB code can be used to extract the numbers on number plate.

CHAPTER 6

CODE (TRAFFIC ESTIMATION)

```
function varargout = traffic_CS(varargin)
% TRAFFIC_CS MATLAB code for traffic_CS.fig
%   TRAFFIC_CS, by itself, creates a new TRAFFIC_CS or raises the existing
%   singleton*.
%
%   H = TRAFFIC_CS returns the handle to a new TRAFFIC_CS or the handle to
%   the existing singleton*.
%
%   TRAFFIC_CS('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in TRAFFIC_CS.M with the given input arguments.
%
%   TRAFFIC_CS('Property','Value',...) creates a new TRAFFIC_CS or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before traffic_CS_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to traffic_CS_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help traffic_CS

% Last Modified by GUIDE v2.5 20-Apr-2015 18:42:33

% Begin initialization code
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
'gui_Singleton', gui_Singleton, ...
'gui_OpeningFcn', @traffic_CS_OpeningFcn, ...
'gui_OutputFcn',  @traffic_CS_OutputFcn, ...
'gui_LayoutFcn',  [], ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code
```

```

% --- Executes just before traffic_CS is made visible.
function traffic_CS_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to traffic_CS (see VARARGIN)

% Choose default command line output for traffic_CS
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes traffic_CS wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = traffic_CS_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
vid=handles.V;

t1=clock;
data=getsnapshot(vid);
data=YUY2toRGB(data);
%h=impoly(data);
%[x y]=CreateMask(data);

diff=imcrop(data,[23 47 179 238]);

diff=im2bw(data,0.3);
diff=imdilate(diff,strel('disk',2));
diff=imerode(diff,strel('line',3,45));

bw=bwlabel(diff,8);
axes(handles.axes2)

```

```

imshow(diff);

axes(handles.axes1)
imshow(data);

stats=regionprops(bw,'BoundingBox','Centroid','Area');
hold on

car=0;
for i=1:length(stats)
f=stats(i).Area;
if (f>100&&f<1000)
bb=stats(i).BoundingBox;
bc=stats(i).Centroid;
rectangle('Position',bb,'EdgeColor','Y','LineWidth',2);
plot(bc(1),bc(2),'-m+');
% a=text(bc(1)+15,bc(2), strcat('X:', num2str(round(bc(1))), 'Y:', num2str(round(bc(2)))));
% set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12, 'Color', 'yellow');
% a=text(bc(1)+15,bc(2)+15, num2str(f));
% set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12, 'Color', 'yellow');
car=car+1;
end
end

if(~isempty(stats))
set(handles.text2,'String',strcat(num2str( car ), ' CARS'));
end

hold off
t2=clock;
time=t2(6)-t1(6);

if(time<0)
time=time+60;
end

set(handles.text3,'String',strcat(num2str( time ),' sec'));
disp(t2-t1);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
vid=handles.V;
stop(vid);
clear all
close all
clc

```

```

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
vid=videoinput('winvideo',2);
% set(vid,'TriggerRepeat',Inf);
% vid.returnedcolorspace='rgb';
% vid.FrameGrabInterval=2;
disp('OK');
handles.V=vid;
guidata(hObject, handles);

```

FUNCTION TO INPUT THE DATA FROM CAMERA AND CONVERT IT INTO RGB

```

function r=takeboxes(NR,container,chk)
%TAKEBOXES helps in determining the values of indices of interested Bounding boxes.
% R=TAKEBOXES(NR,CONTAINER,CHK) outputs the value of indices corresponding
% the desired Bounding boxes. NR is the numberofregionsx4 matrix of all the
% regions' Bounding boxes. CONTAINER is the width of the bin that contain
% all the six bounding boxes of interest. CHK will determine whether
% bounding boxes are y-dimension width's wise grouped or y-coordinate wise
% grouped. CHK=2 considers y-dimension width grouping and CHK=1 considers
% y-coordinate grouping.

takethisbox=[];      % Initialize the variable to an empty matrix.
for i=1:size(NR,1)
if NR(i,(2*chk))>=container(1) && NR(i,(2*chk))<=container(2) % If Bounding box is among the
                                                                container plus tolerance.
takethisbox=cat(1,takethisbox,NR(i,:));      % Take that box and concatenate along first dimension.
end
end
r=[];
for k=1:size(takethisbox,1)
var=find(takethisbox(k,1)==reshape(NR(:,1),1,[])); %Finding the indices of the interested boxes
                                                    among NR
if length(var)==1
                                                    % since x-coordinate of the boxes will be unique.
r=[r var];
else
                                                    % In case if x-coordinate is not unique
for v=1:length(var)
                                                    % then check which box fall under container condition.
M(v)=NR(var(v),(2*chk))>=container(1) && NR(var(v),(2*chk))<=container(2);
end
var=var(M);
r=[r var];
end
end
end

```

ROAD SURVEILLANCE (NUMBER PLATE EXTRACTION)

```
function numberPlateExtractionimresizeimresize
%NUMBERPLATEEXTRACTION extracts the characters from the input number plate image.

f=imread('ABH3.jpg');
f=imresize(f,[400 NaN]);
g=rgb2gray(f);
g=medfilt2(g,[3 3]);
se=strel('disk',1);
gi=imdilate(g,se);
ge=imerode(g,se);
gdif=imsubtract(gi,ge);
gdif=mat2gray(gdif);
gdif=conv2(gdif,[1 1;1 1]);
gdif=imadjust(gdif,[0.5 0.7],[0 1],0.1); B=logical(gdif);
er=imerode(B,strel('line',50,0));
out1=imsubtract(B,er);
F=imfill(out1,'holes');
H=bwmorph(F,'thin',1);
H=imerode(H,strel('line',3,90));
final=bwareaopen(H,100);
% final=bwlabel(final);
Iprops=regionprops(final,'BoundingBox','Image');
NR=cat(1,Iprops.BoundingBox);
r=controlling(NR);
if ~isempty(r)
I={ Iprops.Image };
noPlate=[];
for v=1:length(r)
N=I{ 1,r(v)};
letter=readLetter(N);
```



```

while letter=='O' || letter=='0'
if v<=3
letter='O';
else
letter='0';
end
break;
end
noPlate=[noPlate letter]; end
fid = fopen('noPlate.txt', 'wt');
fprintf(fid,'%s\n',noPlate);
fclose(fid);
winopen('noPlate.txt')

% Uncomment the portion of code below if Database is to be organized. Since our
% project requires database so we have written this code. DB is the .mat
% file containing the array of structure of all entries of database.
% load DB
% for x=1:length(DB)
% recordplate=getfield(DB,{1,x},'PlateNumber');
% if strcmp(noPlate,recordplate)
%     disp(DB(x));
%     disp('*-*-*-*-*-*-*');
% end
% end

else % If fail to extract the indexes in 'r' this line of error will be displayed.
fprintf('Unable to extract the characters from the number plate.\n');
fprintf('The characters on the number plate might not be clear or touching with each other or
boundries.\n');
end
end

```

APPENDIX

- **Image Dilation:** $IM2 = \text{imdilate}(IM, SE)$ dilates the grayscale, binary, or packed binary image IM , returning the dilated image, $IM2$. The argument SE is a structuring element object, or array of structuring element objects, returned by the `strel` function.
- **Image Erosion:** $IM2 = \text{imerode}(IM, SE)$ erodes the grayscale, binary, or packed binary image IM , returning the eroded image $IM2$. The argument SE is a structuring element object or array of structuring element objects returned by the `strel` function.
- **Getsnapshot:** $\text{frame} = \text{getsnapshot}(obj)$ immediately returns one single image frame, frame , from the video input object obj . The frame of data returned is independent of the video input object's `FramesPerTrigger` property and has no effect on the value of the `FramesAvailable` or `FramesAcquired` property. The object obj must be a 1-by-1 video input object.
- **Imcrop:** $I = \text{imcrop}$ creates an interactive Crop Image tool associated with the image displayed in the current figure, called the target image.
- **Im2bw:** $BW = \text{im2bw}(I, \text{level})$ converts the grayscale image I to a binary image. The output image BW replaces all pixels in the input image with luminance greater than level with the value 1 (white) and replaces all other pixels with the value 0 (black).
- **Bwlabel:** $L = \text{bwlabel}(BW)$ returns the label matrix L that contains labels for the 8-connected objects found in BW . The label matrix, L , is the same size as BW .
- **Axes:** `axes` creates an axes graphics object in the current figure using default property values.
- **Stats:** $\text{stats} = \text{regionprops}(BW, \text{properties})$ returns measurements for the set of properties specified by `properties` for each connected component (object) in the binary image, BW . `stats` is a struct array containing a struct for each object in the image.
- **Imread:** $A = \text{imread}(\text{filename})$ reads the image from the file specified by `filename`, inferring the format of the file from its contents. If `filename` is a multi-image file, `imread` reads the first image in the file.
- **Imresize:** $B = \text{imresize}(A, \text{scale})$ returns image B that is scale times the size of A . The input image A can be a grayscale, RGB, or binary image. If scale is between 0 and 1.0, B is smaller than A . If scale is greater than 1.0, B is larger than A . By default, `imresize` uses bicubic interpolation.
- **Rgb2gray:** $I = \text{rgb2gray}(RGB)$ converts the truecolor image RGB to the grayscale intensity image I . The `rgb2gray` function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

- **Medfilt:** $B = \text{medfilt2}(A)$ performs median filtering of the matrix A in two dimensions. Each output pixel contains the median value in a 3-by-3 neighborhood around the corresponding pixel in the input image. `medfilt2` pads the image with 0's on the edges, so the median values for points within one-half the width of the neighborhood ($(m\ n)/2$) of the edges might appear distorted.
- **Strel:** $SE = \text{strel}(\text{shape}, \text{parameters})$ creates a structuring element, SE , of the type specified by `shape`. This table lists all the supported shapes. Depending on `shape`, `strel` can take additional parameters.
- **Imsubtract:** $Z = \text{imsubtract}(X,Y)$ subtracts each element in array Y from the corresponding element in array X and returns the difference in the corresponding element of the output array Z . X and Y are real, nonsparse numeric arrays of the same size and class, or Y is a double scalar. The array returned, Z , has the same size and class as X unless X is logical, in which case Z is double.
- **Mat2gray:** $I = \text{mat2gray}(A, [\text{amin}\ \text{amax}])$ converts the matrix A to the intensity image I . The returned matrix I contains values in the range 0.0 (black) to 1.0 (full intensity or white). `amin` and `amax` are the values in A that correspond to 0.0 and 1.0 in I . Values less than `amin` become 0.0, and values greater than `amax` become 1.0.
- **Conv2:** $C = \text{conv2}(A,B)$ computes the two-dimensional convolution of matrices A and B . If one of these matrices describes a two-dimensional finite impulse response (FIR) filter, the other matrix is filtered in two dimensions. The size of C is determined as follows: if $[m_a, n_a] = \text{size}(A)$, $[m_b, n_b] = \text{size}(B)$, and $[m_c, n_c] = \text{size}(C)$, then $m_c = \max([m_a + m_b - 1, m_a, m_b])$ and $n_c = \max([n_a + n_b - 1, n_a, n_b])$.
- **Imadjust:** $J = \text{imadjust}(I)$ maps the intensity values in grayscale image I to new values in J such that 1% of data is saturated at low and high intensities of I . This increases the contrast of the output image J .
- **Bwmorph:** $BW2 = \text{bwmorph}(BW, \text{operation})$ applies a specific morphological operation to the binary image BW .
- **Imfill:** $W2 = \text{imfill}(BW)$ displays the binary image BW on the screen and lets you define the region to fill by selecting points interactively with the mouse. To use this syntax, BW must be a 2-D image.

REFERENCES

1. Digital Image Processing Using Matlab By Rafael C Gonzalez.
2. Real Time Traffic Density Count Using Image Processing By NaeemAbbas , Md. And M.Tahir(Sir Syed University of Engg and Technology).
3. Automatic Traffic Estimation Using Image Processing By PejmanNiksaz
4. B. Bhanu, Automatic Target Recognition: State of the Art Survey, IEEE Transactions on Aerospace and Electronics Systems, AES-22 (1986) 364-379.
5. Sezgin, R. Tasaltin, A new dichotomization technique to multilevel thresholding devoted to inspection applications, Pattern Recognition Letters, 21 (2000) 151-161.
6. R. Danescu, S. Nedevschi, M. Meinecke and T. Graf, "Stereo Vision Based Vehicle Tracking in Urban Traffic Environments", Intelligent Transportation System, IEEE conference on, (2007), pp. 400-404.
7. N. J. Ferrier, S. M. Rowe, A. Blake, "Real-time traffic monitoring", proceeding of the second IEEE workshop on applications of computer vision", (1994)
8. Sanchez, A. Suarez, P. Conci, A. Nunes, E. Universided, R. J. Carlos, Mostoles, "Video-based distance traffic analysis: Application to vehicle tracking and counting", IEEE, (2010) December 3.
9. J. Melo, A. Naftel, A. Bernardino and J. Santos-Victor, "Detection and classification of highway lanes using vehicle motion trajectories", IEEE Transactions on Intelligent Transportation Systems, vol. 7, no. 2, (2006),
10. Yk. Wang and Sh. Chen, "Robust vehicle detection approach", proc. IEEE conference on advanced video and signal based surveillance 2005,[SI]:IEEE press
11. <http://nl.mathworks.com/help/matlab/ref/imread.html>

IMAGE REFERENCES

1. FIG .1 4X5 Binary Image [10]
2. Fig.2 Binary Image used to test Fitting and Hitting of Structuring Elements S1 and S2. [12]
3. FIG .3 Flowchart [24]
4. FIG .4 Quantum QHM500-8LM(S) WEB CAMERA [25]
5. FIG .5 DUMMY ROAD [27]
6. Reference image with no traffic (FIG. 6) [30]
7. Results obtained when there are vehicles present on the road. (FIG 7) [30]
8. Road with one vehicle detected(FIG. 8) [31]
9. Road with three vehicles detected (FIG.9) [31]
10. Road with four vehicles detected. (FIG 10) [32]
11. INPUT IMAGE FOR NUMBER PLATE EXTRACTION: (FIG 11) [32]
12. OUTPUT OBTAINED: (FIG 12) [33]
13. FIG 12(a) [33]
14. Second input for vehicle's number plate detection: (FIG 13) [34]
15. Output obtained: Fig 14 [34]