

A  
Dissertation  
On

## **Real Time Vehicle Number Plate Recognition System**

Submitted in Partial fulfillment of the requirement  
for the award of Degree of

**MASTER OF ENGINEERING**  
**(Electronics & Communication Engineering)**

Submitted By:

**MANPREET SINGH**  
College Roll No: 20/E&C/08  
University Roll No: 8583

Under the Guidance of:  
**Mr. RAJESH ROHILLA**  
(ASSISTANT PROFESSOR)  
Dept. of Electronics & Communication  
Delhi Technological University  
[formerly Delhi College of Engineering]



**DEPARTMENT OF ELECTRONICS & COMMUNICATION  
ENGINEERING  
DELHI COLLEGE OF ENGINEERING  
DELHI UNIVERSITY  
2008-2010**

# **CERTIFICATE**



**DELHI COLLEGE OF ENGINEERING**  
(Govt. of National Capital Territory of Delhi)  
BAWANA ROAD, DELHI – 110042

Date: \_\_\_\_\_

---

---

This is to certify that the work contained in this dissertation entitled “**Real Time Vehicle Number Plate Recognition System**” submitted by **Manpreet Singh**, University Roll No- 8583 in the requirement for the partial fulfillment for the award of the degree of **Master of Engineering in Electronics & Communication Engineering**, Delhi College of Engineering is an account of his work carried out under my guidance and supervision in the academic year 2009-2010.

The work embodied in this major project has not been submitted to any other Institute/University for the award of any other degree to the best of my knowledge.

---

---

**RAJESH ROHILLA**  
( Project Guide )  
Assistant Professor  
Department of Electronics and Communication.  
Delhi College of Engineering,  
University of Delhi, India

# ACKNOWLEDGEMENT

---

---

It is a great pleasure to have the opportunity to extend my heartiest felt gratitude to everybody who helped me throughout the course of this project.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisor **Mr. Rajesh Rohilla** Assistant Professor, Department of Electronics & Communication, Delhi College of Engineering Delhi, for his invaluable guidance, encouragement and patient reviews. I am very thankful to **Prof. Rajiv Kapoor**, H.O.D Department of Electronics & Communication, Delhi College of Engineering Delhi, and **Prof. Asok Bhattacharyya**, former H.O.D Electronics & Communication Department who allows me to do project under the Guidance of Mr. Rajesh Rohilla on Image Processing. With their continuous inspiration, valuable guidance in carrying out this work under his effective supervision, encouragement, enlightenment and cooperation, it becomes possible to complete this dissertation and all of them kept on boosting me with time, to put an extra ounce of effort to realize this work.

I would also like to take this opportunity to present my sincere regards to all the faculty members of the Department for their support and encouragement.

I am grateful to my parents for their moral support all the time; they have been always around to cheer me up, in the odd times of this work. I am also thankful to my classmates for their unconditional support and motivation during this work.

## **MANPREET SINGH**

M.E. (Electronics & Communication Engineering)  
College Roll No. 20/E&C/08  
University Roll No. 8583  
Department of Electronics & Communication Engineering  
Delhi College of Engineering, Delhi-42

# ABSTRACT

---

---

The process of vehicle number plate recognition requires a very high degree of accuracy when we are working on a very busy road or parking which may not be possible manually as a human being tends to get fatigued due to monotonous nature of the job and they cannot keep track of the vehicles when there are multiple vehicles are passing in a very short time .To overcome this problem, many efforts have been made by the researchers across the globe for last many years.

This thesis develops an algorithm for Real Time Number Plate Recognition System based on computer vision techniques in a real-time mode invoke precious and complicated demand of efficient computer algorithms and technological solutions. Real time analysis then leads to generate reports of vehicle which passed through monitored area.

A Real Time Number Plate Recognition System is one kind of an Intelligent Transport System and is of considerable interest because of its potential applications in highway electronic toll collection and traffic monitoring systems. This type of applications puts high demands on the reliability of a Real Time Number Plate Recognition System. A lot of work has been done regarding Real Time Number Plate Recognition systems for Korean, Chinese, European and US license plates that generated many commercial products. However, little work has been done for Indian license plate recognition systems. The purpose of this thesis was to develop a real time application which recognizes Number plates from cars at a gate, for example at the entrance of a parking area or a border crossing. The system, based on regular PC with camera, catches frames which include a visible car number plate and processes them. Once a number

plate is detected, its digits are recognized, displayed on the User Interface or checked against a database. The focus is on the design of algorithms used for extracting the number plate from a single image, isolating the characters of the plate and identifying the individual characters. The Proposed system has been implemented using Vision Assistant 8.5 & LabVIEW 8.5. The performance of the system has been investigated on real images of about 100 vehicles.

# Organization of Thesis

**Chapter 1:-** The first chapter briefly reviews the literature and the previous work done.

**Chapter 2:-** The second chapter gives a brief introduction to the system elements its applications, working and structure of proposed system.

**Chapter 3:-** The third chapter gives a detailed description of analysis and processing tools available in application software Vision Assistant 8.5, on which our work is focused.

**Chapter 4:-** The fourth chapter is focused on the Camera Optics for improving image quality.

**Chapter 5:-** The fifth chapter discusses the implementation of various tools of application software for simulation and testing part of thesis.

**Chapter 6:-** The sixth chapter gives the problem definition and proposed solution.

**Chapter 7:-** The seventh chepter discusses the results obtained after testing.

**Finally, concluding thesis in sixth chapter with future scope.**

# Chapter 1

## Literature Review

### 1.1 Introduction

Number plate recognition systems have received a lot of attention from the research community. Much research has been done on Korean, Chinese, and English number plates. A distinctive feature of research work in this area is being restricted to a specific region, city, or country. This is due to the lack of standardization among different number plates (i.e., the dimension and the layout of the number plates). This section gives an overview of the research carried out so far in this area and the techniques employed in developing an RTNPR system in lieu of the following four stages: image acquisition, number plate extraction, number plate segmentation and number plate recognition phases. In the next section various existing or novel methods for the image acquisition phase are presented.

### 1.2 Image Acquisition

Image Acquisition is the first step in an RTNPR system and there are a number of ways to acquire images, the current literature discusses different image acquisition methods used by various authors. Yan et. al. [20] used an image acquisition card that converts video signals to digital images based on some hardware-based image preprocessing. Naito et. al. [13,14,16] developed a sensing system, which uses two CCDs (Charge Coupled Devices) and a prism to split an incident ray into two lights with different intensities. The main feature of this sensing system is that it covers wide illumination conditions from twilight to noon under sunshine, and this system is capable of capturing images of fast moving vehicles without blurring. Salgado et. al. [15] used a Sensor subsystem having a high resolution CCD camera supplemented with a number of new digital operation capabilities. Kim et. al. [17] uses a video camera to acquire the image. Comelli et. al. [6] used a TV camera and a frame grabber card to acquire the image for the developed vehicle LPR system.

### 1.3 Number Plate Extraction

Number plate extraction is the most important phase in an RTNPR system. This section discusses some of the previous work done during the extraction phase. Hontani et. al. [21] proposed a method for extracting characters without prior knowledge of their position and size in the image. The technique is based on scale shape analysis, which in turn is based on the assumption that, characters have line-type shapes locally and blob-type shapes globally. In the scale shape analysis, Gaussian filters at various scales blur the given image and larger size shapes appear at larger scales. To detect these scales the idea of principal curvature plane is introduced. By means of normalized principal curvatures, characteristic points are extracted from the scale space  $x$ - $y$ - $t$ . The position  $(x, y)$  indicates the position of the figure and the scale  $t$  indicates the inherent characteristic size of corresponding figures. All these characteristic points enable the extraction of the figure from the given image that has line-type shapes locally and blob-type shapes globally. Kim et. al. [17] used two Neural Network-based filters and a post processor to combine two filtered images in order to locate the license plates. The two Neural Networks used are vertical and horizontal filters, which examine small windows of vertical and horizontal cross sections of an image and decide whether each window contains a license plate. Cross-sections have sufficient information for distinguishing a plate from the background. Lee et. al. [5] and Park et. al. [11] devised a method to extract Korean license plate depending on the color of the plate. A Korean license plate is composed of two different colors, one for characters and other for background and depending on this they are divided into three categories. In this method a neural network is used for extracting color of a pixel by HLS (Hue, Lightness and Saturation) values of eight neighboring pixels and a node of maximum value is chosen as a representative color. After every pixel of input image is converted into one of the four groups, horizontal and vertical histogram of white, red and green (i.e. Korean plates contains white, red and green colors) are calculated to extract a plate region. To select a probable plate region horizontal to vertical ratio of plate is used. Dong et. al [10] presented histogram based approach for the extraction phase. Kim G. M [9] used Hough transform for the extraction of the license plate. The algorithm behind the method consists of five steps. The first step is to threshold the gray scale source image, which leads to a binary image. Then in the second stage the resulting image is passed through two parallel sequences, in order to extract horizontal and vertical line segments respectively. The result is an image with edges highlighted. In the third step the



resultant image is then used as input to the Hough transform, this produces a list of lines in the form of accumulator cells. In fourth step, the above cells are then analyzed and line segments are computed. Finally the list of horizontal and vertical line segments is combined and any rectangular regions matching the dimensions of a license plate are kept as candidate regions. The disadvantage is that, this method requires huge memory and is computationally expensive.

## **1.4 Segmentation**

This section discusses previous work done for the segmentation of characters. Many different approaches have been proposed in the literature and some of them are as follows, Nieuwoudt et. al. [8] used region growing for segmentation of characters. The basic idea behind region growing is to identify one or more criteria that are characteristic for the desired region. After establishing the criteria, the image is searched for any pixels that fulfill the requirements. Whenever such a pixel is encountered, its neighbors are checked, and if any of the neighbors also match the criteria, both the pixels are considered as belonging to the same region. Morel et. al. [7] used partial differential equations (PDE) based technique, Neural network and fuzzy logic were adopted in for segmentation into individual characters.

## **1.5 Recognition**

This section presents the methods that were used to classify and then recognize the individual characters. The classification is based on the extracted features. These features are then classified using either the statistical, syntactic or neural approaches. Some of the previous work in the classification and recognition of characters is as follows, Hasen et. al. [23] discusses a statistical pattern recognition approach for recognition but their technique found to be inefficient. This approach is based on the probabilistic model and uses statistical pattern recognition approach. Cowell et. al. [24] discussed the recognition of individual Arabic and Latin characters. Their approach identifies the characters based on the number of black pixel rows and columns of the character and comparison of those values to a set of templates or signatures in the database. Cowell et. al. [22] discusses the thinning of Arabic characters to extract essential structural information of each character which may be later used for the classification stage. Mei Yu et. al.

[18] and Naito et. al. [12] used template matching. Template matching involves the use of a database of characters or templates. There is a separate template for each possible input character. Recognition is achieved by comparing the current input character to each of template in order to find the one which matches the best. If  $I(x,y)$  is the input character,  $T_n(x,y)$  is template  $n$ , then the matching function  $s(I,T_n)$  will return a value indicating how well template  $n$  matches the input. Hamami et. al. [25] adopted a structural or syntactic approach to recognize characters in a text document, this technique can yield a better result when applied on the recognition of individual characters. This approach is based on the detection of holes and concavities in the four directions (up, down, left and right), which permits the classification of characters into different classes. In addition, secondary characteristics are used in order to differentiate between the characters of each class. The approaches discussed in this paragraph are based on the structural information of the characters and uses syntactic pattern recognition approach. Hu [1] proposed seven moment that can be used as features to classify the characters. These moments are invariant to scaling, rotation and translation. The obtained moments acts as the features, which are passed to the neural network for the classification or recognition of characters. Zernike moments have also been used by several authors [4,2,3] for recognition of characters. Using zernike moments both the rotation variant and rotation invariant features can be extracted. These features then uses neural network for the recognition phase. Neural network accepts any set of distinguishable features of a pattern as input. It then trains the network using the input data and the training algorithms to recognize the input pattern (In this case characters).

## 1.6 Summary

This chapter reviewed material relevant to the license plate recognition system. The relevant techniques used in the four phases of an LPR system were discussed. In the case of image acquisition, a sensing system using two Charge Coupled Devices along with a prism gives better input to the system. Because the main feature of this sensing system is that it covers wide illumination conditions from twilight to noon under sunshine, and this system is capable of capturing images of fast moving vehicles without blurring video camera with a frame. In the case of license plate extraction, Hough transform was used to extract the license plate by using storing the horizontal and vertical edge information. But the disadvantage is that, this method requires huge memory and is computationally expensive. Various segmentation techniques were presented in the segmentation stage. Then the literature for recognition of characters using various approaches was also discussed.

# Chapter 2

## Introduction

### 2.1 Introduction

Massive integration of information technologies into all aspects of modern life caused demand for processing vehicles as conceptual resources in information systems. Because a standalone information system without any data has no sense, there was also a need to transform information about vehicles between the reality and information systems. This can be achieved by a human agent, or by special intelligent equipment which is able to recognize vehicles by their number plates in a real environment and reflect it into conceptual resources. Because of this, various recognition techniques have been developed and number plate recognition systems are today used in various traffic and security applications, such as parking, access and border control, or tracking of stolen cars.

This thesis presents a number plate recognition system as an application of computer vision. Computer vision is a process of using a computer to extract high level information from a digital image. This chapter will set the scene by first presenting some applications of a number plate recognition system. Next, we discuss the elements that are commonly used in a number plate recognition system. Following this, the working of a typical number plate recognition system is described. Next, we present the structure of proposed license plate recognition system. Finally, the objectives of the work are stated. The chapter ends with a brief overview of the rest of this thesis.

### 2.2 Applications of NPR Systems

Vehicle number plate recognition is one form of automatic vehicle identification system. NPR systems are of considerable interest, because of their potential applications to areas such as highway electronic toll collection, automatic parking attendant, petrol station forecourt surveillance, speed limit enforcement, security, customer identification enabling personalized services, etc. Real time NPR plays a major role in automatic monitoring of traffic rules and maintaining law enforcement on public roads. This area is challenging because it requires an

integration of many computer vision problem solvers, which include Object Detection and Character Recognition. The automatic identification of vehicles by the contents of their number plates is important in private transport applications. There are many applications of such recognition systems, some of them are discussed below.

**Law Enforcement** :- The plate number is used to produce a violation fine on speeding vehicles, illegal use of bus lanes, and detection of stolen or wanted vehicles. License plate recognition technology has gained popularity in security and traffic applications as it is based on the fact that all vehicles have a number plate and there is no need to install any additional tracking apparatus. The main advantage is that the system can store the image record for future references. The rear part of the vehicle is extracted off the filmed image and is given to the system for processing. The processed result is fed into the database as input. The violators can pay the fine online and can be presented with the image of the car as a proof along with the speeding information.

**Parking** :- The NPR system is used to automatically enter pre-paid members and calculate parking fee for non-members (by comparing the exit and entry times). The car plate is recognized and stored and upon its exit the car plate is read again and the driver is charged for the duration of parking.

**Automatic Toll Gates** :- Manual toll gates require the vehicle to stop and the driver to pay an appropriate tariff. In an automatic system the vehicle would no longer need to stop. As it passes the toll gate, it would be automatically classified in order to calculate the correct tariff.

**Border Crossing** :- This application assists the registry of entry or exits to a country, and can be used to monitor the border crossings. Each vehicle information is registered into a central database and can be linked to additional information.

**Homeland Security** :- The NPR system's ability to read strings of alpha-numeric characters and compare them instantaneously to Hot Lists allows a Command Center to organize and strategize efforts in reaction to the information captured. Fixed LPR systems, which can be mounted to bridges, gates and other high traffic areas, can help keep a tight watch on entire cities, ports,

borders and other vulnerable areas. Every NPR camera is capturing critical data such as color photos, date and time stamps, as well as GPS coordinates on every vehicle that passes or is passed. This incredible database provides a wealth of clues and proof, which can greatly aid Law Enforcement with

- Pattern recognition
- Placing a suspect at a scene
- Watch list development
- Identifying witnesses
- Possible visual clues revealed within the image of a car's immediate environment

## 2.3 Elements of Typical Number Plate Recognition System

Number Plate Recognition Systems normally consist of the following units:

**Camera** :- Takes image of a vehicle from either front or rear end.

**Illumination** :- A controlled light that can bright up the plate, and allows day and night operation. In most cases the illumination is Infra-Red (IR) which is invisible to the driver.

**Frame Grabber** :- An interface board between the camera and the PC that allows the software to read the image information.

**Computer** :- Normally a PC running Windows or Linux. It runs the Number Plate Recognition System application that controls the system, reads the images, analyzes and identifies the plate, and interfaces with other applications and systems.

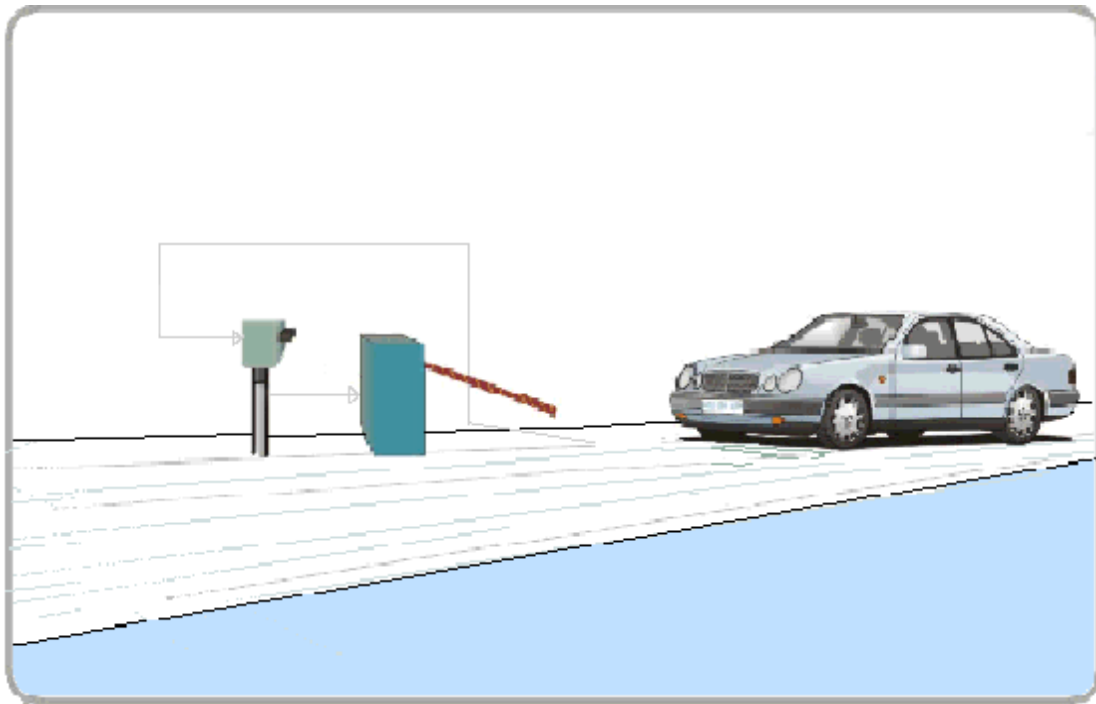
**Software** :- The application and the recognition package.

**Hardware** :- Various input/output boards used to interface the external world (such as control boards and networking boards).

**Database** :- The events are recorded on a local database or transmitted over the network. The data includes the recognition results and (optionally) the vehicle or driver-face image file.

## 2.4 Working of Typical LPR System

When the vehicle approaches the secured area, the RTNPR unit senses the car and activates the illumination (invisible infra-red in most cases) as shown in Figure below. The LPR unit takes the pictures from either the front or rear plates from the RTNPR camera. The image of the vehicle contains the number plate. The RTNPR unit feeds the input image to the system. The system then enhances the image, detects the plate position, extracts the plate, segments the characters on the plate and recognizes the segmented characters, Checks if the vehicle appears on a predefined list of authorized vehicles, If found, it signals to open the gate by activating its relay. The unit can also switch on a green "go-ahead" light or red "stop" light. The unit can also display a welcome message or a message with personalized data. The authorized vehicle enters into the secured area. After passing the gate its detector closes the gate. Now the system waits for the next vehicle to approach the secured area.



**Figure 2.1 A car approaching a Number Plate Recognition System**

## **2.5 Structure of the Proposed System**

The system presented is designed to recognize license plates from the front and rear of the vehicle. Input to the system is an image sequence acquired by a digital camera that consists of a license plate and its output is the recognition of characters on the license plate. The system consists of the standard four main modules in an number plate recognition system, viz. Image acquisition, License plate extraction, License plate segmentation and License plate recognition. The first task acquires the image. The second task extracts the region that contains the license plate. The third task isolates the characters, letters and numerals (total of 10 digits), as in the case of Indian License Plates. The last task identifies or recognizes the segmented characters.

### **2.5.1 Image Acquisition**

This is the first phase in an number plate recognition system. This phase deals with acquiring an image by an acquisition method. In our proposed system, we used a high resolution digital camera to acquire the input image. The input image is 620 x 480 pixels.

### **2.5.2 Number Plate Extraction**

Number Plate Extraction is a key step in an LPR system, which influences the accuracy of the system significantly. This phase extracts the region of interest, i.e., the license plate, from the acquired image. The proposed approach involves “Masking of a region with high probability of license plate and then scanning the whole masked region for license plate”.

### **2.5.3 Number Plate Segmentation**

Number Plate Segmentation, which is sometimes referred to as Character Isolation takes the region of interest and attempts to divide it into individual characters. In the proposed system segmentation is done in the OCR section which will be described in next chapters.

### **2.5.4 Number Plate Recognition**

The last phase in NPR system is to recognize the isolated characters. After splitting the extracted license plate into individual character images, the character in each image can be identified. There are many methods used to recognize isolated characters. In the proposed system we are using



Optical Character Recognition which is an inbuilt feature in Vision Assistant 8.5. Optical Character Recognition is described in detail in next chapters.

## **2.6 Objective**

The work presented here aims at the following aspects:

- Study the existing license plate recognition systems,
- Develop a new technique or enhance existing techniques for each phase in a number plate recognition system,
- Compare the various techniques at hand with the proposed system, and
- Build a system that delivers optimal performance both in terms of speed and accuracy.

# Chapter 3

## Software Development

### 3.1 Digital Images

This section contains information about the properties of digital images, image types, file formats, the internal representation of images in IMAQ Vision, image borders, and image masks.

#### 3.1.1 Definition of a Digital Image

An image is a 2D array of values representing light intensity. For the purposes of image processing, the term image refers to a digital image. An image is a function of the light intensity  $F(x,y)$  where  $f$  is the brightness of the point  $(x, y)$ , and  $x$  and  $y$  represent the spatial coordinates of a picture element, or pixel. By convention, the spatial reference of the pixel with the coordinates  $(0, 0)$  is located at the top, left corner of the image. Notice in Figure 3.1 that the value of  $x$  increases moving from left to right, and the value of  $y$  increases from top to bottom.

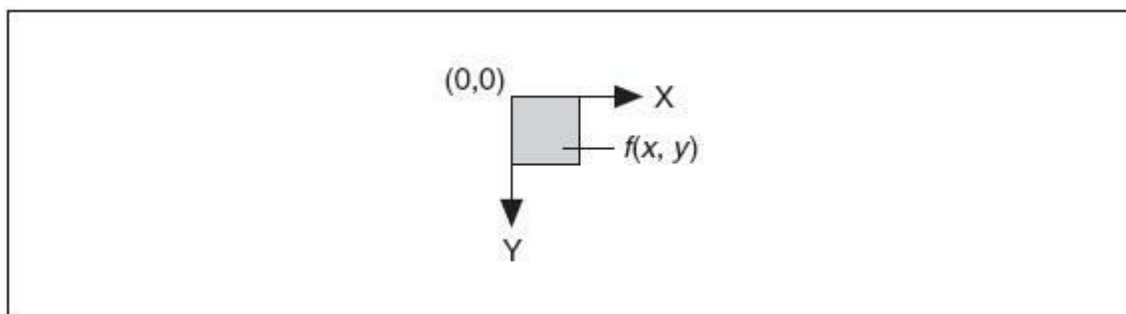


Fig 3.1 Special reference of the  $(0,0)$  pixel.

**An image is of one of the following image data types:**

- 8-bit (default)
- 16-bit
- Float
- Complex
- RGB
- HSL

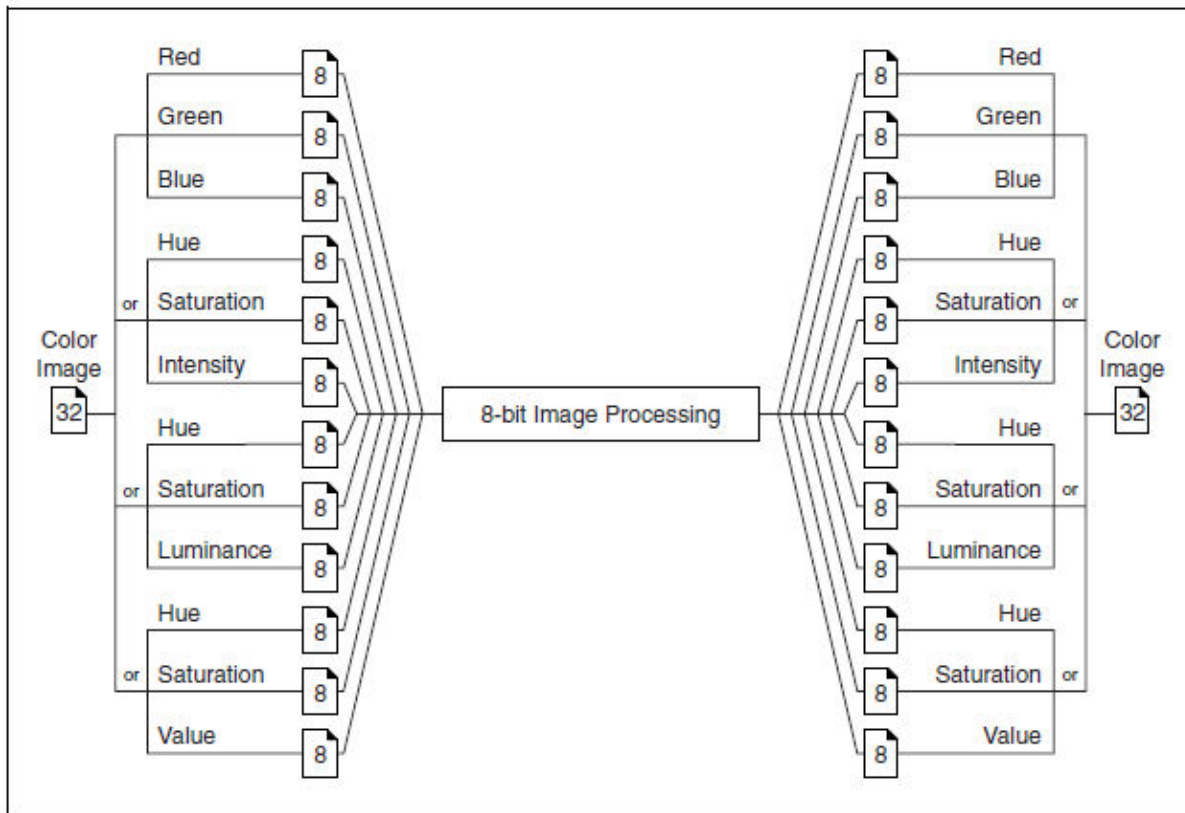


Figure 3.2 Primary Components of a Color Image

## 3.2 Vision Assistant: An overview

A detailed overview of vision assistant 8.5 is given as under.

### 3.2.1 Acquiring Images

Vision Assistant offers three types of image acquisitions: snap, grab, and sequence. A snap acquires and displays a single image. A grab acquires and displays a continuous set of images, which is useful while focusing the camera. A sequence acquires images according to settings that are specified and sends the images to the Image Browser. Using Vision Assistant, images can be acquired with various National Instruments digital and analog IMAQ devices. Vision Assistant provides specific support for several Sony, JAI, and IEEE 1394 cameras. IMAQ devices can be configured in National Instruments Measurement & Automation Explorer (MAX). The sequence can be stopped at any frame, capture the image, and send the image to the Image Browser for processing.

## **(A) Opening the Acquisition window**

Complete the following steps to acquire images.

1. Click **Start » Programs » National Instruments » Vision Assistant 8.5**.
2. Click **Acquire Image** in the Welcome screen to view the Acquisition functions. If Vision Assistant is already running, click the **Acquire Image** button in the toolbar. We must have one of the following device and driver software combinations to acquire live images in Vision Assistant.
  - National Instruments IMAQ device and NI-IMAQ 3.0 or later
  - IEEE 1394 industrial camera and NI-IMAQ for IEEE 1394 Cameras 1.5 or later
  - USB Webcam etc.

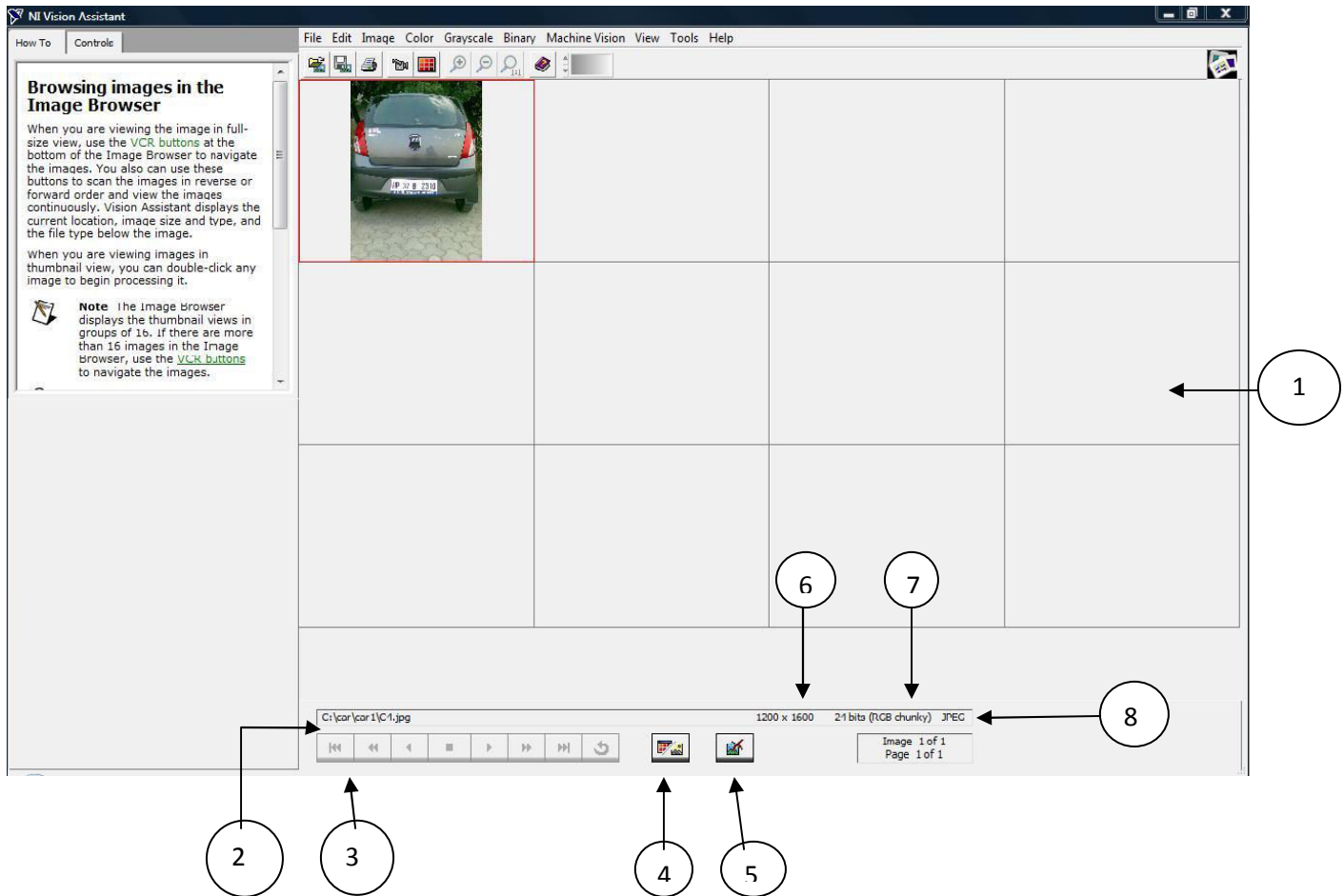
Click **Acquire Image**. The Parameter window displays the IMAQ devices and channels installed on the computer.

## **(B) Snapping an image**

1. Click **File » Acquire Image**.
2. Click **Acquire Image** in the Acquisition function list.
3. Select the appropriate device and channel.
4. Click the **Acquire Single Image** button to acquire a single image with the IMAQ device and display it.
5. Click the **Store Acquired Image in Browser** button to send the image to the Image Browser.
6. Click **Close** to exit the Parameter window.
7. Process the image in Vision Assistant.

### 3.2.2 Managing Images

1. Select **Start » Programs » National Instruments » Vision Assistant 8.5**
2. To load images, click **Open Image** in the Welcome screen.



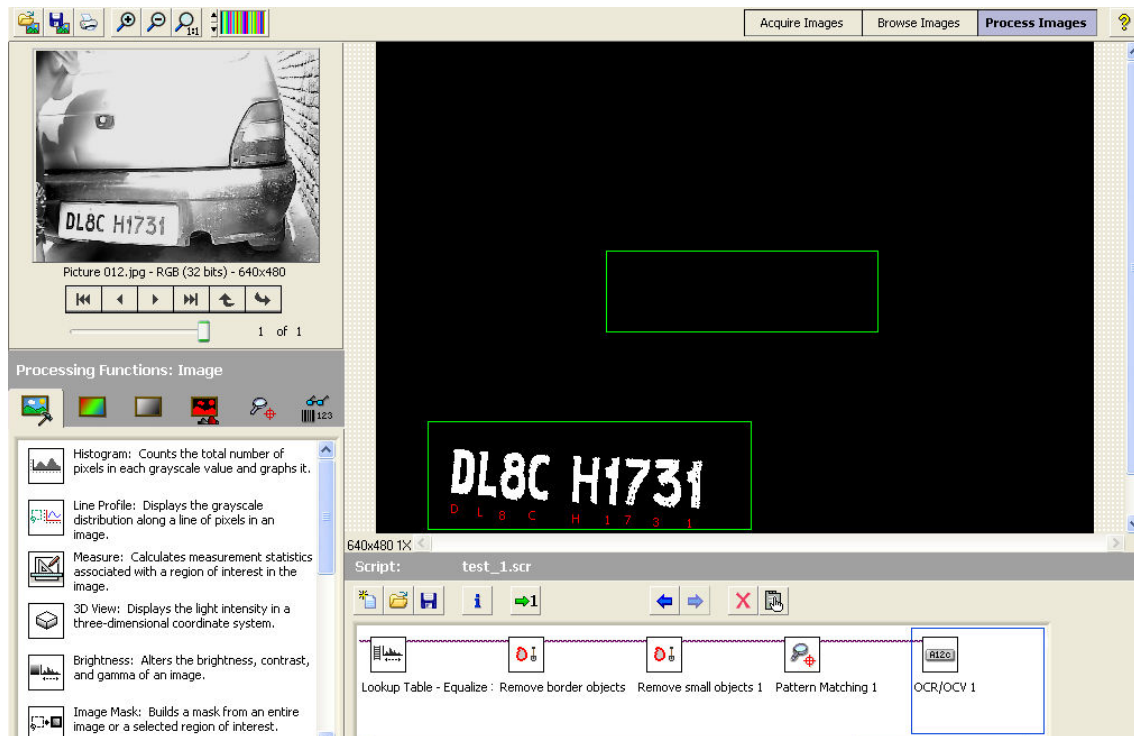
- |                          |                                       |                       |
|--------------------------|---------------------------------------|-----------------------|
| <b>1.</b> Image Browser  | <b>4.</b> Thumbnail/ Full-Size Toggle | <b>7.</b> Image Type  |
| <b>2.</b> Image Location | <b>5.</b> Close Selected Image(s)     | <b>8.</b> File Format |
| <b>3.</b> Browse Buttons | <b>6.</b> Image Size                  |                       |

**Fig: 3.2 Image Browser**

3. Navigate to select the image which we want to process. If analysis is to be done on more than one image then there is **Select All Files** option also available in Vision Assistant. It previews the images in the **Preview Image** window and displays information about the file type.
4. Click **OK**. Vision Assistant loads the image files into the **Image Browser**, as shown in Figure 3.2. The **Image Browser** provides information about the selected image such as image size,

location, and type. We can view new images in either thumbnail view, as shown in Figure 3.2 or in full-size view, which shows a single full-size view of the selected image.

5. Click the **Thumbnail/Full-Size View Toggle** button to view the first image in full size.



**Fig: 3.3 Processing an Image**

6. Double-click on one of the selected images to begin processing it. Vision Assistant loads the image into the Processing window, as shown in Figure 3.3.

7. Now, image is ready for processing as per requirements. We can apply various functions on image that are available in processing function window as shown in figure given above. These processing steps get recorded in the Script window. The script records the processing operations and all its parameters. If we want to run the same operation on other images, we can save the script and use it again.

8. Select **File » Save Script** and name the script.

9. To run script on other images follow the steps given below:

- (i) Load the image. (ii) Select **File » Open Script**
- (iii) Click the **Run Once** button in the script window

10. Select **File » Exit** to close Vision Assistant.

### 3.2.3 Image Processing Functions

The various functions available in Vision Assistant that can be used for image processing and analysis are listed below. The following gives an overview of available functions in Vision Assistant 7.1.

**(i) Image analysis functions.** Vision Assistant provides the following image analysis functions:-

- ***Histogram*** counts the total number of pixels in each grayscale value and graphs the result.
- ***Line Profile*** returns the grayscale values of the pixels along a line that is drawn with the Line Tool from the Tools palette and graphs the result
- ***Measure*** calculates measurement statistics associated with a region of interest in the image.
- ***3D View*** displays an image using an isometric view.
- ***Image Mask*** builds a mask from an entire image or region of interest.
- ***Geometry*** modifies the geometrical representation of an image.
- ***Image Buffer*** stores and retrieves images from buffers.
- ***Get Image*** opens a new image from a file.

**(ii) Colour image processing functions.** Vision Assistant provides the following set of functions for processing and analyzing colour images:

- ***Color Operators*** applies an arithmetic operation between two images or between an image and a constant .
- ***Extract Color Planes*** extracts the Red, Green, or Blue (RGB) plane or the Hue, Saturation, or Luminance (HSL) plane of a color image.
- ***Color Threshold*** applies a threshold to the three planes of an RGB or HSL image.
- ***Color Location*** locates colors in an image.
- ***Color Matching*** compares the color content of one or multiple regions in an image to a reference color set.
- ***Color Pattern Matching*** searches for a color template in an image.

**(iii) Grayscale image processing and analysis functions.** Vision Assistant provides the following functions for grayscale image processing and analysis:

- **Lookup Table** applies predefined lookup table transformations to the image to modify the dynamic intensity of regions in the image with poor contrast.
- **Filters** include functions for smoothing, edge detection, and convolution.
- **Gray Morphology** modifies the shape of objects in grayscale images using erosion, dilation, opening, and closing functions.
- **FFT Filters** applies a frequency filter to the image.
- **Threshold** isolates the pixels we specify and sets the remaining pixels as background pixels.
- **Operators** perform basic arithmetic and logical operations between two images or between an image and a constant.
- **Conversion** converts the current image to the specified image type.
- **Centroid** computes the energy center of a grayscale image or an area of interest on a grayscale image.

**(iv) Binary processing and analysis functions.** Vision Assistant provides the following functions for binary processing and analysis:

- **Basic Morphology** performs morphology transformations that modify the shape of objects in binary images.
- **Adv. Morphology** performs high-level operations on particles in binary images.
- **Particle Filter** filters objects based on shape measurements.
- **Invert Binary Image** reverses the dynamic of an image that contains two different grayscale populations.
- **Shape Matching** finds image objects that are shaped like the object specified by the template.



- **Particle Analysis** computes more than 80 measurements on objects in an image, including the area and perimeter of the objects.
- **Circle Detection** separates overlapping circular objects and classifies them according to their radii.

**(v) Machine vision functions.**

Vision Assistant provides the following machine vision functions:

- **Edge Detection** finds edges along a line that we draw with the Line Tool.
- **Find Straight Edge** finds points along the edge of an object and then finds a line describing the edge.
- **Find Circular Edge** locates the intersection points between a set of search lines within a circular area, or annulus, and then finds the best fit circle.
- **Clamp** finds edges within a rectangular region of interest (ROI) drawn in the image and measures the distance between the first and last edge.
- **Pattern Matching** locates regions of a grayscale image that match a predetermined template. Pattern Matching can find template matches regardless of poor lighting, blur, noise, shifting of the template, and rotation of the template.
- **Caliper** computes measurements such as distances, areas, and angles—based on results returned from other machine vision and image processing functions.

## 3.3 Script Development

Once the desired image is loaded in the Processing window as shown in Figure 5.3, we follow the steps discussed below to develop the script.

### 3.3.1 Extracting color planes from image

Since the color information is redundant so we extract color plane from the acquired 32-bit colored image to make it an 8-bit grayscale image.

1. Click **Color » Extract Color Plane** or select **Extract Color Plane** in the **Color** tab of the Processing Functions Palette.
2. Select the **color plane** to be extracted.

3. Click **OK** to add this step to the script.

The choice of color plane to be extracted is made by trying all of them one by one and selecting the one which gives the best image thereafter. We have selected to extract value plane from the image. Figure 5.4 shows how the image looks after applying this function.

### 3.3.2 Store Image to Buffer

Buffer is a storing element which stores the image temporary and we can retrieve the image from the buffer at any instance.

### 3.3.3 Image Filtering

Filter your image when you need to improve the sharpness of transitions in the image or increase the overall signal-to-noise ratio of the image. You can choose either a lowpass or highpass filter depending on your needs.

Lowpass filters remove insignificant details by smoothing the image, removing sharp details, and smoothing the edges between the objects and the background. You can use the IMAQ LowPass (**Grayscale »Filters**) or define your own lowpass filter with the IMAQ Convolute or IMAQ NthOrder.

Highpass filters emphasize details, such as edges, object boundaries, or cracks. These details represent sharp transitions in intensity value. You can define your own highpass filter with IMAQ Convolute or IMAQ NthOrder or use the IMAQ EdgeDetection or IMAQ CannyEdgeDetection (**Grayscale»Filters**). IMAQ EdgeDetection allows you to find edges in an image using predefined edge detection kernels, such as the Sobel, Prewitt, and Roberts kernels.

As the word *filtering* suggests, most of the methods described in this section are used to improve the quality of the image. In general, these methods calculate new pixel values by using the original pixel value and those of its neighbors. For example, we can use the equation

$$s_{\text{out}}(x, y) = \frac{1}{m^2} \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} s_{\text{in}}(x + k - u, y + k - v) \cdot f(u, v) \quad (3.1)$$

which looks very complicated. Let's start from the left: Eq. (3.1) calculates new pixel values  $s_{\text{out}}$  by using an  $m \times m$  array. Usually,  $m$  is 3, 5, or 7; in our case we use  $m = 3$ , which means that the original pixel value  $s_{\text{in}}$  and the values of the eight surrounding pixels are used to calculate the new value.  $k$  is defined as  $k = (m - 1)/2$ .

The values of these nine pixels are modified with a *filter kernel*

$$\mathbf{F} = (f(u, v)) = \begin{pmatrix} f(0, 0) & f(0, 1) & f(0, 2) \\ f(1, 0) & f(1, 1) & f(1, 2) \\ f(2, 0) & f(2, 1) & f(2, 2) \end{pmatrix} \quad (3.2)$$

As you can see in Eq. (4.12), the indices  $u$  and  $v$  depend upon  $x$  and  $y$  with  $k = (m - 1)/2$ . The filter kernel moves from the top-left to the bottom-right corner of the image. The process described by Eq. (4.12) is also called *convolution*; that is why the filter kernel may be called *convolution kernel*.

#### A) Convolution Filter

IMAQ Convolute (**Image Processing»Filters**) allows you to use a predefined set of lowpass and highpass filters. Each filter is defined by a kernel of coefficients. Use the IMAQ GetKernel VI (**Image Processing» Filters**) to retrieve predefined kernels. If the predefined kernels do not meet your needs, define your own custom filter using a LabVIEW 2D array of floating point numbers.

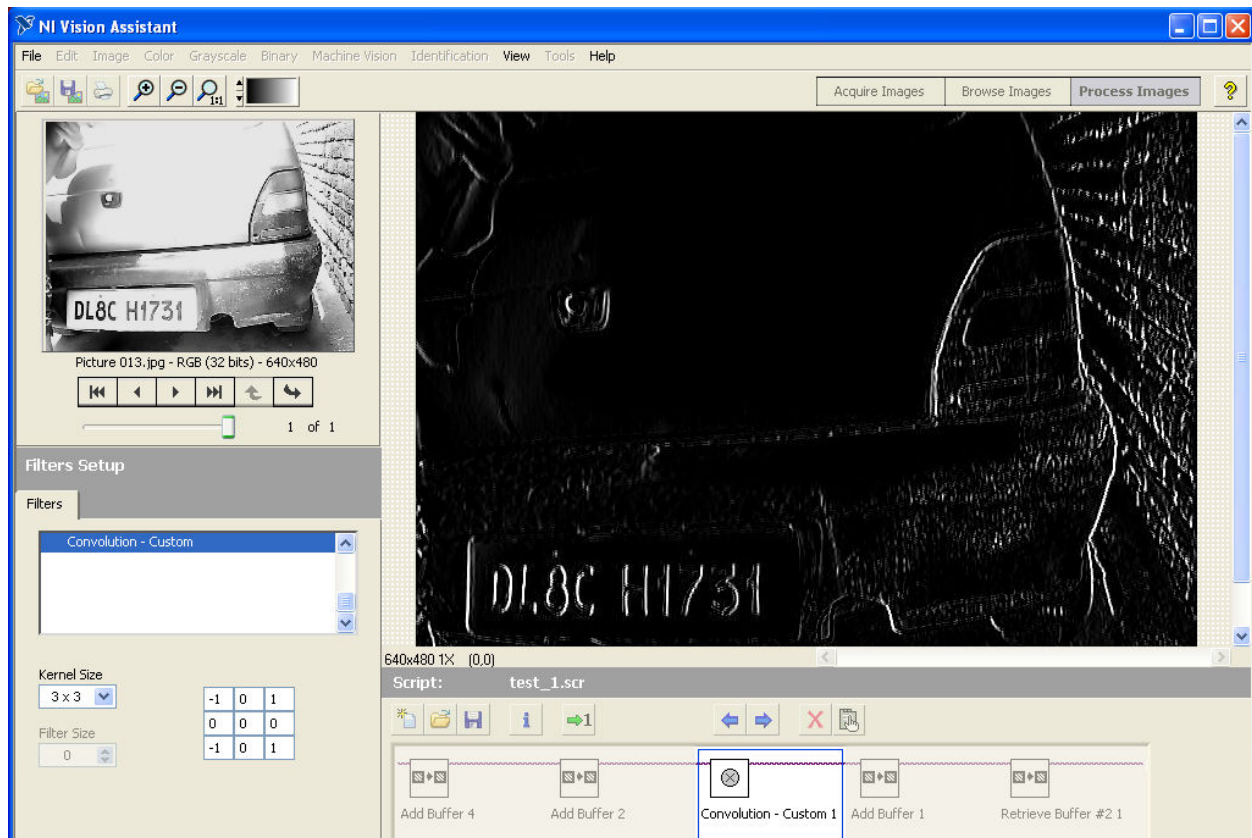
#### B) Nth Order Filter

IMAQ NthOrder (**Image Processing»Filters**) allows you to define a lowpass or highpass filter depending on the value of  $N$  that you choose. One specific Nth order filter, the median filter, removes speckle noise, which appears as small black and white dots.

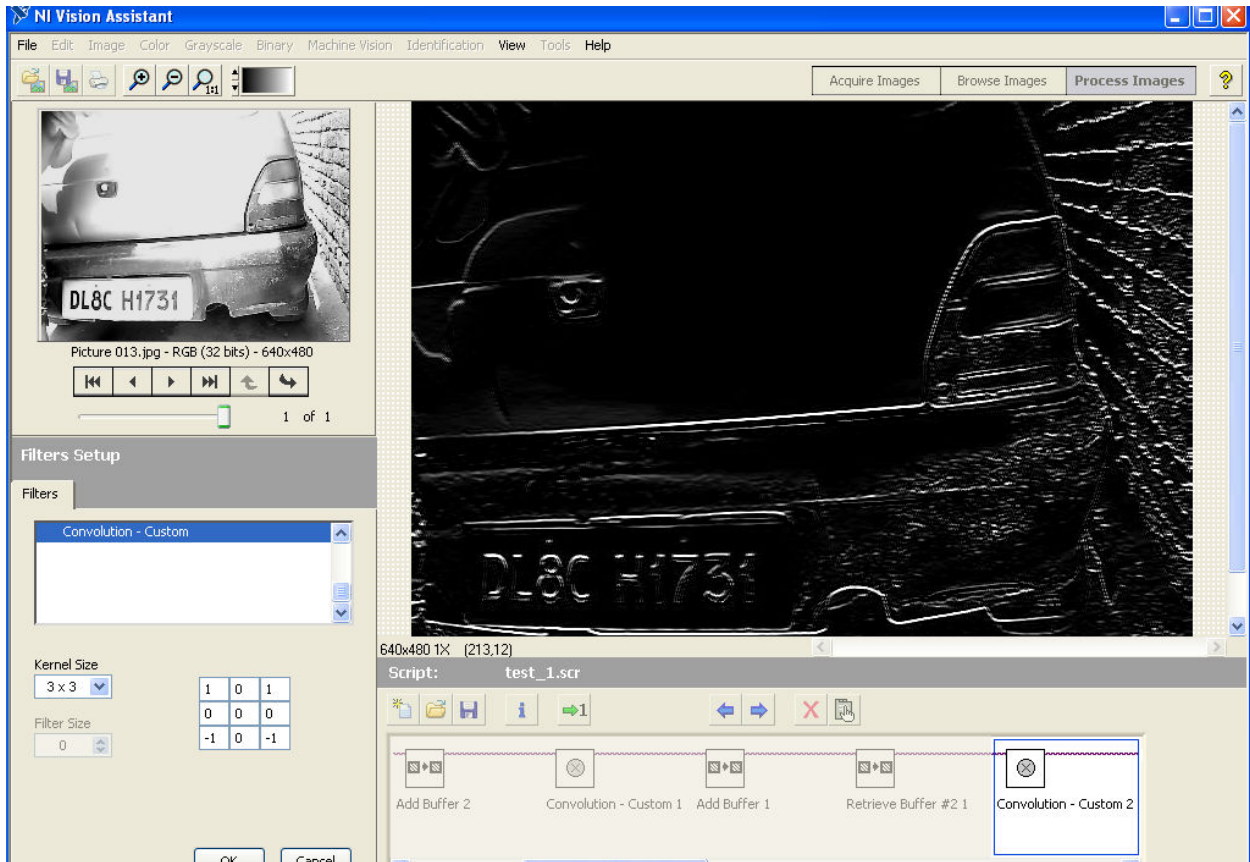
## Edge Detection and Enhancement

An important area of study is the detection of edges in an image. An *edge* is an area of an image in which a significant change of pixel brightness occurs.

**Filter Families: Gradient.** A gradient filter extracts a significant brightness change in a specific direction and is thus able to extract edges rectangular to this direction. Figure 3.4 shows the effect of a vertical gradient filter, which extracts dark-bright changes from left to right.



**Fig. 3.4 Filter Example: vertical gradient filter.**



**Fig: 3.5 Filter Example: Horizontal gradient filter.**

Figure 3.5 shows the similar effect in a horizontal direction (dark-bright changes from bottom to top). Gradient kernels like those in Figure 3.4 and Figure 3.5 are also known as *Prewitt filters* or *Prewitt kernels*. Another group of gradient kernels, for example,

$$\mathbf{F} = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

are called *Sobel filters* or *Sobel kernels*. A Sobel kernel gives the specified filter direction a stronger weight. In general, the mathematic value *gradient* specifies the amount of change of a value in a certain direction.

Note that the center coefficient of this kernel is 0; if we make it equal to 1, the edge information is added to the original value. The original image remains and the edges in the specified direction are highlighted.

### 3.3.4 Thresholding

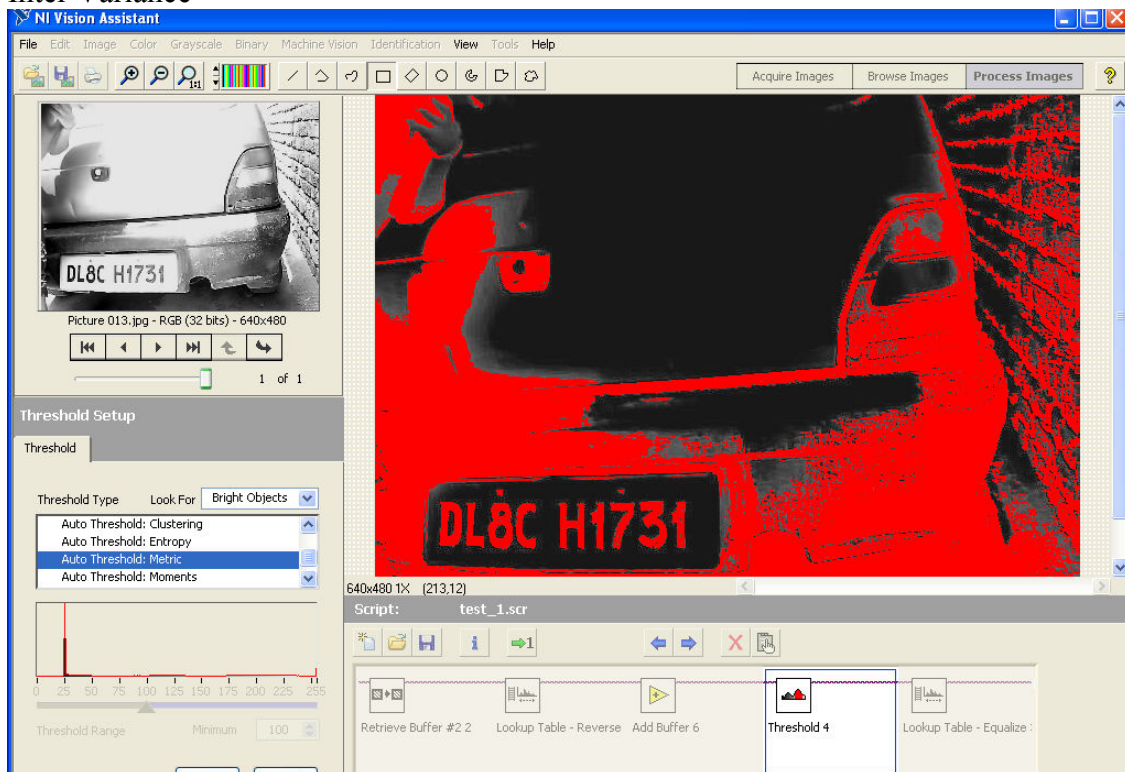
Thresholding is one of the most important concepts in the image process. Thresholding is separating image pixels into foreground and background pixels based on their intensity values. Foreground pixels are those whose intensity values are within the lower and upper threshold values of the threshold range. Background pixels are those whose intensity values lie outside the lower and upper threshold values of the threshold range. Vision Development Module includes one manual method, two local methods and five automatic methods of calculating the thresholding range:

Local Thresholding Methods are:

- 1) NIBlack
- 2) Background Correction.

Auto Thresholding Methods are:

- 1) Clustering
- 2) Entropy
- 3) Matric
- 4) Moment
- 5) Inter Variance



**Fig: 3.6 Image after Thresholding**

### 3.3.5 Improve an Image

Using the information you gathered from analyzing your image, you may want to improve the quality of your image for inspection. You can improve your image with lookup tables, filters, grayscale morphology, and Fast Fourier transforms.

#### Lookup Tables

Apply lookup table (LUT) transformations to highlight image details in areas containing significant information at the expense of other areas. A LUT transformation converts input grayscale values in the source image into other grayscale values in the transformed image. IMAQ Vision provides four VIs that directly or indirectly apply lookup tables to images:

- IMAQ MathLookup (**Image Processing»Grayscale»Lookup Table**)—Converts the pixel values of an image by replacing them with values from a predefined lookup table. IMAQ Vision has seven predefined lookup tables based on mathematical transformations.
- IMAQ UserLookup (**Image Processing»Grayscale»Lookup Table**)—Converts the pixel values of an image by replacing them with values from a user-defined lookup table.
- IMAQ Equalize (**Image Processing»Grayscale»Lookup Table**)—Distributes the grayscale values evenly within a given grayscale range. Use IMAQ Equalize to increase the contrast in images containing few grayscale values.
- IMAQ Inverse (**Image Processing»Grayscale»Lookup Table**)—Inverts the pixel intensities of an image to compute the negative of the image. For example, use IMAQ Inverse before applying an automatic threshold to your image if the background pixels are brighter than the object pixels.

### 3.3.6 Improve the Binary Image

After you threshold your image, you may want to improve the resulting binary image with binary morphological functions. You can use primary *binary morphology* or advanced binary morphology to remove unwanted particles, separate connected particles, or improve the shape of particles. Primary morphology functions work on the image as a whole by processing pixels individually. Advanced morphology operations are built upon the primary morphological operators and work on particles as a whole as opposed to individual pixels.



The advanced morphology functions require that you specify the type of *connectivity* to use. Connectivity specifies how IMAQ Vision determines whether two adjacent pixels belong to the same particle. Use *connectivity-4* when you want IMAQ Vision to consider pixels to be part of the same particle only when the pixels touch along an adjacent edge. Use *connectivity-8* when you want IMAQ Vision to consider pixels to be part of the same particle even if the pixels touch only at a corner.

## 1) Remove Unwanted Particles

Use the IMAQ RejectBorder VI (**Image Processing»Morphology**) to remove particles that touch the border of the image. Reject particles on the border of the image when you suspect that the information about those particles is incomplete. Use the IMAQ RemoveParticle VI (**Image Processing»Morphology**) to remove large or small particles that do not interest you. You also can use the Erode, Open, and POpen functions in the IMAQ Morphology VI (**Image Processing»Morphology**) to remove small particles. Unlike the IMAQ RemoveParticle VI, these three functions alter the size and shape of the remaining particles.

Use the hit-miss function of the IMAQ Morphology VI to locate particular configurations of pixels, which you define with a structuring element. Depending on the configuration of the structuring element, the hit-miss function can locate single isolated pixels, cross-shape or longitudinal patterns, right angles along the edges of particles, and other user-specified shapes.

## 2) Separate Touching Particles

Use the IMAQ Separation VI (**Image Processing»Morphology**) or apply an erosion or an open function with the IMAQ Morphology VI to separate touching objects. IMAQ Separation is an advanced VI that separates

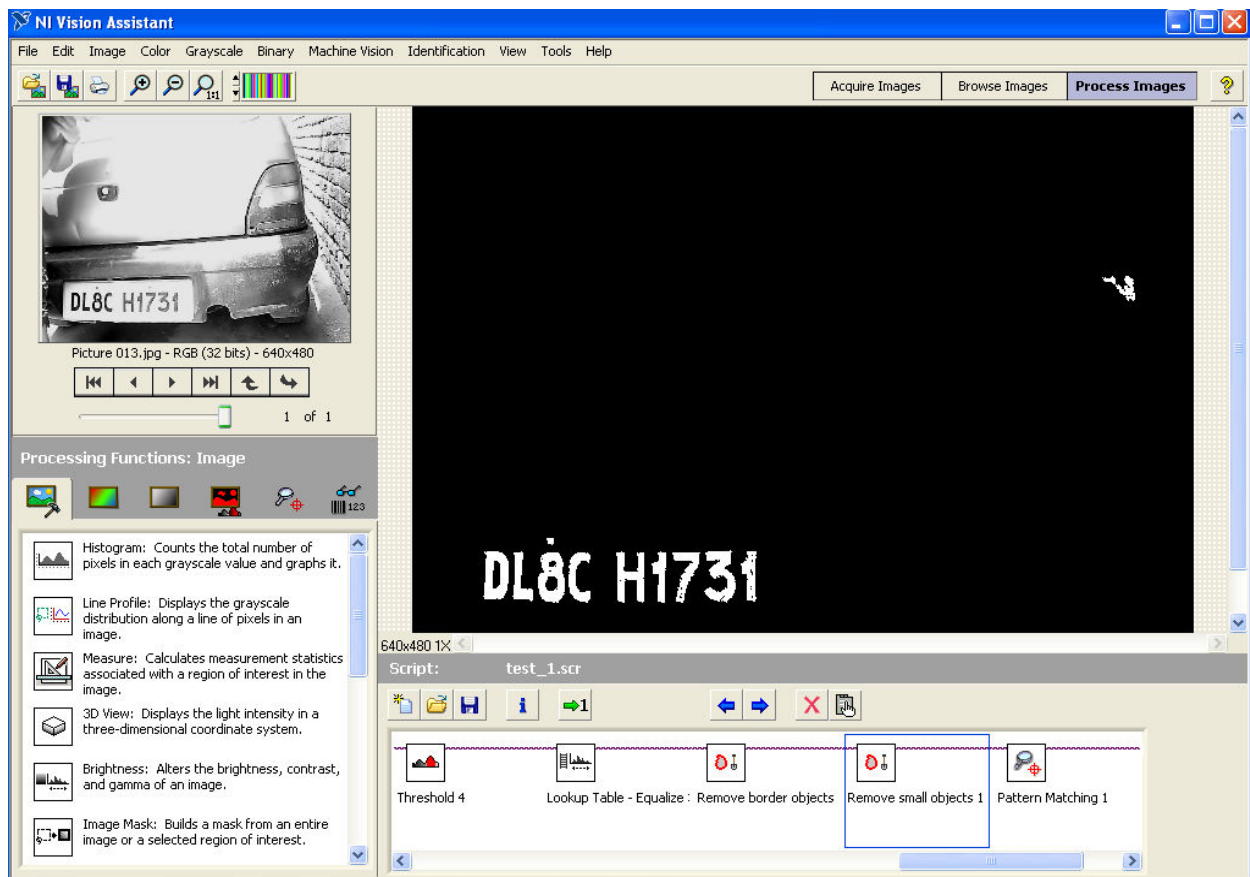
particles without modifying their shapes. However, erosion and open operations alter the shape of all the particles.

**Note** A separation is a time-intensive operation compared to an erosion or open operation. Consider using an erosion if speed is an issue in your application.



### 3) Improve Particle Shapes

Use the IMAQ FillHoles VI (**Image Processing»Morphology**) to fill holes in the particles. Use the IMAQ Morphology VI (**Image Processing» Morphology**) to perform a variety of operations on the particles. You can use the Open, Close, POpen, PClose, and AutoM operations to smooth the boundaries of the particles. Open and POpen smooth the boundaries of the particle by removing small isthmuses while Close widens the isthmuses. Close and PClose fill small holes in the particle. AutoM removes isthmuses and fills holes.



**Fig: 3.7 Image after Pre-processing**

### 3.3.7 Pattern Matching

Pattern matching is a method of identifying features in an image that match a smaller template image (that is, the "pattern" to be matched). The process involves two phases: an off-line learning phase in which the template is processed, and a matching phase that can be executed in real time. This document explains some of the factors involved in pattern matching performance, and strategies that you can use to achieve the best results.

#### Principles of Pattern Matching

Let us start by assuming that we have taken two images ( $I_1$  and  $I_2$ ), separated by a time distance of  $\Delta t = 0.0012s$ . We subsequently divide both images into smaller regions, also known as sub-windows, interrogation-windows or interrogation-regions. We then compare each sub-window in the first image with the corresponding sub-window in the second image. We shall

hereafter let  $I_1^{i,j}$  denote sub-window number  $i,j$  in the first image and  $I_2^{i,j}$  the corresponding sub-window in the second image. We now aim to see if we can identify a displacement of the

pattern in  $I_1^{i,j}$ . To do this we can evaluate the squared Euclidean distance between the two sub-windows. This is defined as

$$R_s(s, t) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [I_1^{i,j}(m, n) - I_2^{i,j}(m - s, n - t)]^2.$$

This means that, for every possible overlap of the sub-windows, we calculate the sum of the squared difference between them. In other words this means that we are looking for the position where the sub-windows are the "least unlike". Let us look in a bit more detail to this simple mathematical formula. If we expand the square parentheses on the right hand side we may get

$$\begin{aligned}
R_{\epsilon}(s, t) &= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [I_1^{i,j}(s, t) - I_2^{i,j}(m - s, n - t)]^2 \\
&= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I_1^{i,j}(m, n)^2 - 2I_1^{i,j}(s, t) \cdot I_2^{i,j}(m - s, n - t) + I_2^{i,j}(m - s, n - t)^2.
\end{aligned} \tag{3.1}$$

We should notice that the first term,  $I_1^{i,j}(m, n)^2$ , is merely a constant since it does not depend

on  $s$  and  $t$ . The last term,  $I_2^{i,j}(m - s, n - t)^2$  is seen to depend on  $s$  and  $t$ , but we notice that it is just dependent on the second image. So to sum up, only the middle term actually deals with both our images and as a matter of fact this term (without the  $-2$ ) is usually referred to as cross-correlation (or circular cross-correlation) and defined as

$$R(s, t) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I_1^{i,j}(m, n) \cdot I_2^{i,j}(m - s, n - t). \tag{3.2}$$

The basic assumption here is that the pattern in  $I_2$  is evenly distributed so that the sum of  $I_2^{i,j}()^2$  does not change as we vary  $s$  and  $t$ .

Traditionally in PIV, equation 3.2 has been preferred, and it is also the basis of many of the different algorithms in **MatPIV** ('single', 'multi' and 'multin' options). The reason for this is primarily that equation 3.2 can be calculated using FFTs (and will therefore execute faster).

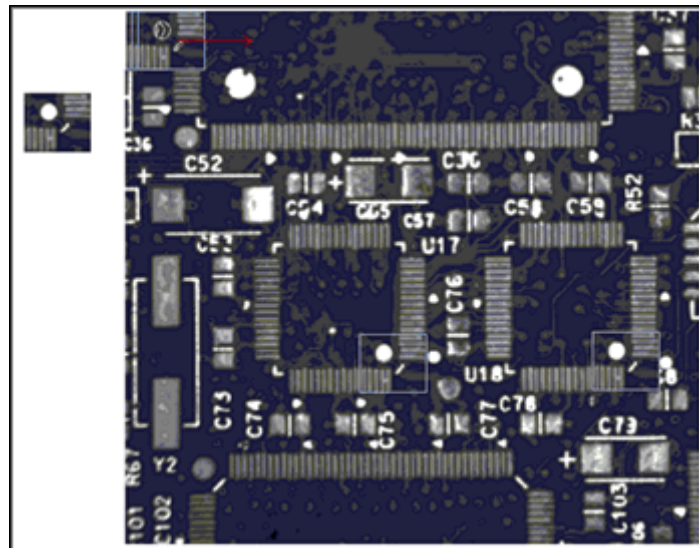
The use of equation 3.1.1 in PIV has primarily been advocated by Gui and Merzkirch (1996) and Gui and Merzkirch (2000) and this is implemented in the **MatPIV** option called 'mqd'.

In the field of pattern matching another approach has often been chosen. Considering the fact that the last term in equation 3.1 may be non-constant, many people choose to apply so called normalized correlation, which is defined as

$$R(s, t) = \frac{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I_1^{i,j}(s, t) \cdot I_2^{i,j}(m-s, n-t)}{[\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I_1^{i,j}(m, n)^2 \cdot \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I_2^{i,j}(m-s, n-t)^2]^{1/2}} \quad (3.3)$$

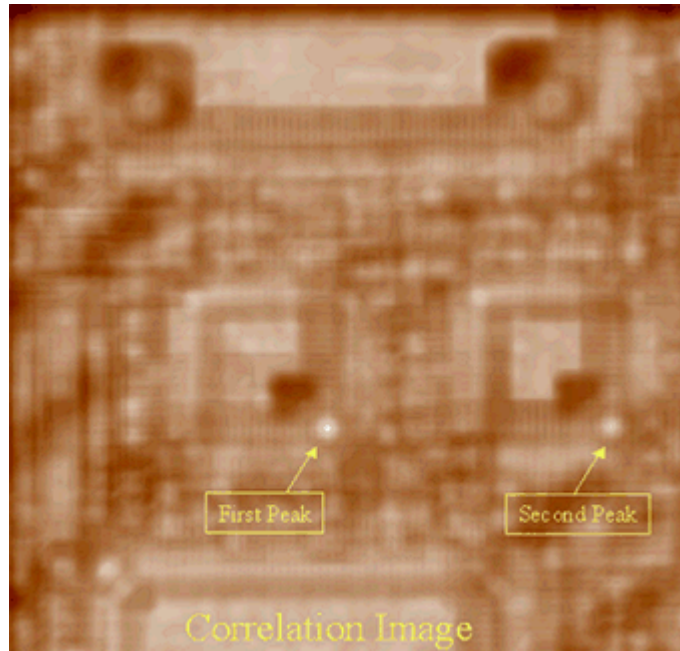
### The Learning Phase

The learning phase of pattern matching involves analyzing the template image to find features that can be exploited for efficient matching performance. This phase is what gives advanced pattern matching techniques dramatically improved performance compared to traditional grayscale correlation methods. The traditional methods have no learning phase; the template is simply compared to every possible location in the image via a 2D correlation. This is very computationally intensive and involves many redundant calculations. Below is an example of a printed circuit board and template. After performing the correlation at every point in the image, two matches were found.



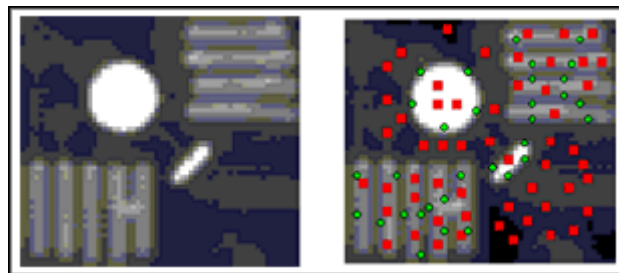
**Fig : 3.8 Image with Template**

The correlation image produced is shown below. The two bright spots correspond to the two matches found:



**Fig: 3.9 Correlation Image**

With a learning phase, you can incorporate sub-sampling of the template image to drastically reduce the number of calculations. Below is an example of a template that has been sub-sampled. The red squares capture information about the overall structure of the image, while the green circles identify features of the image that will be useful for localizing the match precisely.



The National Instruments pattern matching algorithm incorporates the following features in the learning phase:

- Pseudo-random sub-sampling: If you sub-sample the template by taking a uniform grid, it is possible to miss important features of the template, such as horizontal and vertical

edges. Random sampling tends to produce clusters and open areas; having clusters of samples in the same area of the template is inefficient, while the unsampled areas may contain important information. The pseudo-random technique maximizes the uniformity of the sampling throughout the template without using a pre-defined grid.

- Stability analysis: The pseudo-random sample pixels are analyzed by checking their surrounding neighborhood for stability (uniformity). Each pixel is classified according to how large its stable neighborhood is (that is, 3x3, 5x5, and so on). This information is helpful in reducing the number of comparisons needed in the matching phase.
- Feature identification: Finally, an edge detection operation is performed on the template image. The resulting edge locations are retained for use in fine-tuning the location of the matched image.
- Rotational-invariant analysis: If the user needs to be able to find a rotated version of the pattern in the search image, the learning phase also obtains information that can be used to detect rotation. This is done by identifying a circular intensity profile in the template image. A rotated version of the template has the same profile, but is shifted left or right by the amount of rotation.

This learning phase is quite complex, and the calculations can take up to several seconds to perform for large or complex templates. This phase only needs to be done once per template, however. The results can be stored in a file for use later in the actual pattern matching operation.

### **Matching Patterns**

The matching phase uses the information from the learning phase to eliminate as much unnecessary calculation as possible. The matching algorithm used depends on whether the user has specified shift-invariant matching (finding the template at any location in the search image) or rotation-invariant matching (finding the template at any location AND rotation in the search image). Both are two-pass processes.

### **Shift-Invariant Matching**

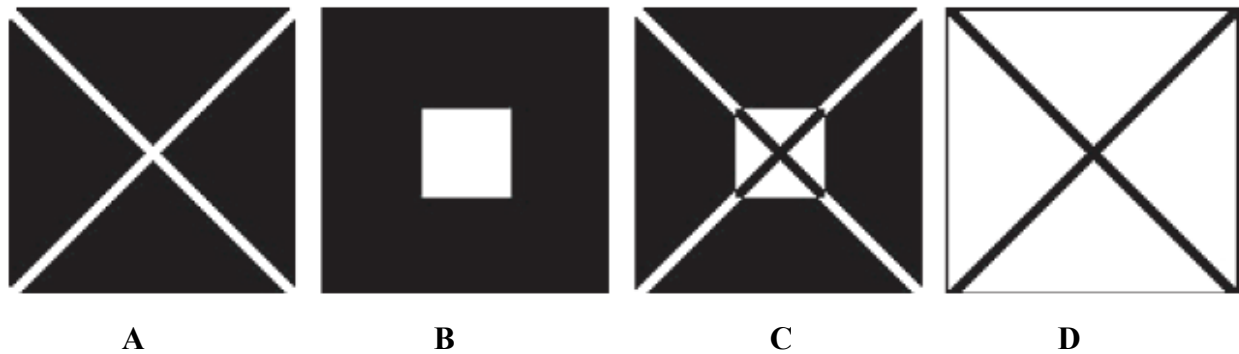
1. The first pass is a correlation that uses only the pseudo-randomly sampled pixels from the template image. The results of the stability analysis are used to determine how many positions in the search image can be skipped without missing any important features. For example, if all the sub-sampled pixels were found to be stable in a 3x3 neighborhood, the matching algorithm can skip two out of three correlations in each row and column while still guaranteeing that a match will be detected. This reduces the number of calculations required by a factor of 9. The first pass produces a number of candidate matches with rough position information.
2. The second pass only operates on the candidates identified in the first pass. The edge detection results of the learning phase are used to fine-tune the location of each match, and a score is produced for each based on the correlation result at that location. A user-provided score threshold determines which candidates are returned as matches.

### **Rotation-Invariant Matching**

1. The first pass uses the circular intensity profile from the learning phase to search for shifted versions of that profile throughout the image. The user can input an allowable rotation range (in degrees) to reduce the number of calculations required in this pass. Several candidate matches are identified in this pass.
2. The second pass uses the pseudo-randomly sampled pixels to perform a correlation with all the candidates. A score is produced for each candidate to determine whether it should be classified as a match or not.

### 3.3.8 Image Mask

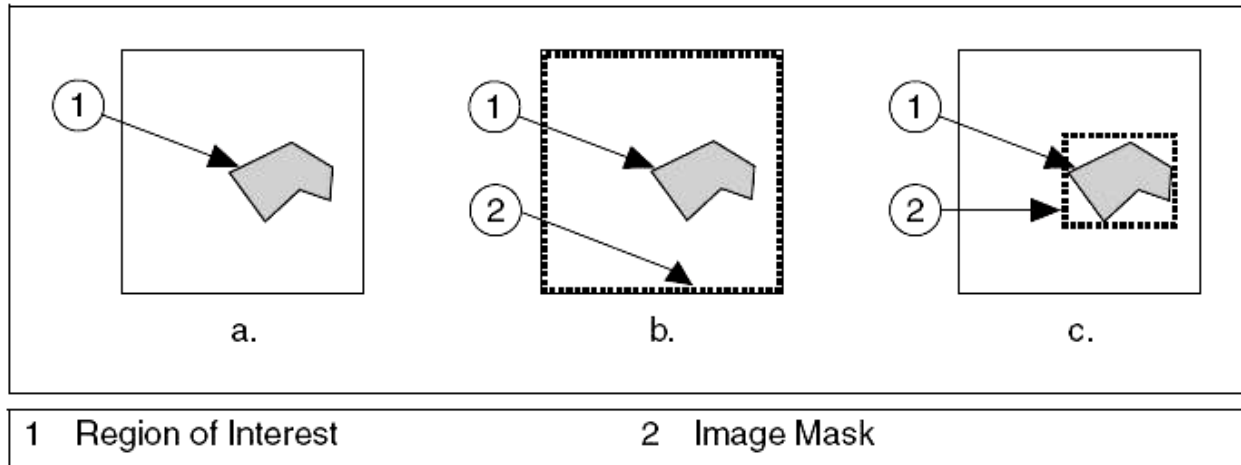
An image mask isolates parts of an image for processing. If a function has an image mask parameter, the function process or analysis depends on both the source image and the image mask. An image mask is an 8-bit binary image that is the same size as or smaller than the inspection image. Pixels in the image mask determine whether corresponding pixels in the inspection image are processed. If a pixel in the image mask has a nonzero value, the corresponding pixel in the inspection image is processed. If a pixel in the image mask has a value of 0, the corresponding pixel in the inspection image is not processed. Pixels in the source image are processed if corresponding pixels in the image mask have values other than zero. A mask affects the output of the function that inverts the pixel values in an image.



**Fig 3.10 Effect of an image mask.**

Figure 3.5A shows the inspection image. Figure 3.5B shows the image mask. Pixels in the mask with zero values are represented in black, and pixels with nonzero values are represented in white. Figure 3.5C shows the inverse of the inspection image using the image mask. Figure 3.5D shows the inverse of the inspection image without the image mask. We can limit the area in which our function applies an image mask to the bounding rectangle of the region we want to process. This technique saves memory by limiting the image mask to only the part of the image containing significant information. To keep track of the location of this region of interest (ROI) in regard to the original image, IMAQ Vision sets an offset. An offset defines the coordinate position in the original image where you want to place the origin of the image mask. Figure 3.6 illustrates the different methods of applying image masks. Figure 3.6a shows the ROI in which you want to apply an image mask.





**Fig 3.11 Using an Offset to limit an image mask.**

Figure 3.6b shows an image mask with the same size as the inspection image. In this case, the offset is set to  $[0, 0]$ . A mask image also can be the size of the bounding rectangle of the ROI, as shown in Figure 3.6c, where the offset specifies the location of the mask image in the reference image. We can define this offset to apply the mask image to different regions in the inspection image.

## **3.4 Optical Character Recognition (OCR)**

The exact mechanisms that allow humans to recognize objects are yet to be understood, but the three basic principles are already well known by scientists – integrity, purposefulness and adaptability. These principles constitute the core of OCR allowing it to replicate natural or human-like recognition. Optical Character Recognition provides machine vision functions we can use in an application to perform OCR. OCR is the process by which the machine vision software reads text and/or characters in an image.

### **3.4.1 What is OCR**

Optical Character Recognition (OCR) is a type of document image analysis where a scanned digital image that contains either machine printed or handwritten script is input into an OCR software engine and translating it into an editable machine readable digital text format (like ASCII text). OCR works by first pre-processing the digital page image into its smallest component parts with layout analysis to find text blocks, sentence/line blocks, word blocks and character blocks. Other features such as lines, graphics, photographs etc are recognised and discarded. The character blocks are then further broken down into components parts, pattern

recognized and compared to the OCR engines large dictionary of characters from various fonts and languages. Once a likely match is made then this is recorded and a set of characters in the word block are recognized until all likely characters have been found for the word block. The word is then compared to the OCR engine's large dictionary of complete words that exist for that language. These factors of characters and words recognised are the key to OCR accuracy by combining them the OCR engine can deliver much higher levels of accuracy. Modern OCR engines extend this accuracy through more sophisticated pre-processing of source digital images and better algorithms for fuzzy matching, sounds-like matching and grammatical measurements to more accurately establish word accuracy.

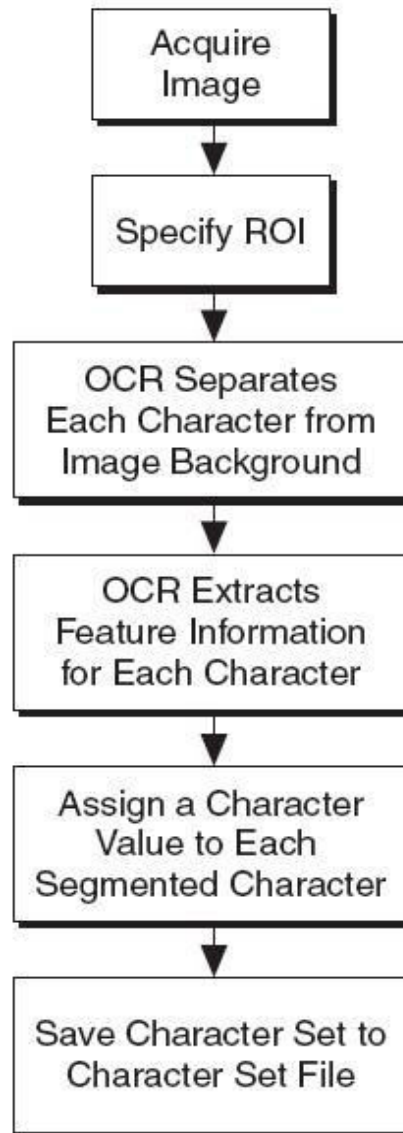
### **3.4.2 When to Use**

Typically, machine vision OCR is used in automated inspection applications to identify or classify components. For example, we can use OCR to detect and analyze the serial number on an automobile engine that is moving along a production line. Using OCR in this instance helps you identify the part quickly, which in turn helps you quickly select the appropriate inspection process for the part. We can use OCR in a wide variety of other machine vision applications, such as the following

- Inspecting pill bottle labels and lot codes in pharmaceutical applications.
- Verifying wafers and IC package codes in semiconductor applications.
- Controlling the quality of stamped machine parts.
- Sorting and tracking mail packages and parcels.
- Reading alphanumeric characters on automotive parts.
- **And of course for license plate recognition applications.**

### **3.4.3 Training Characters**

Training involves teaching OCR the characters and/or patterns you want to detect during the reading procedure. Figure 3.7 illustrates the steps involved in the training procedure.

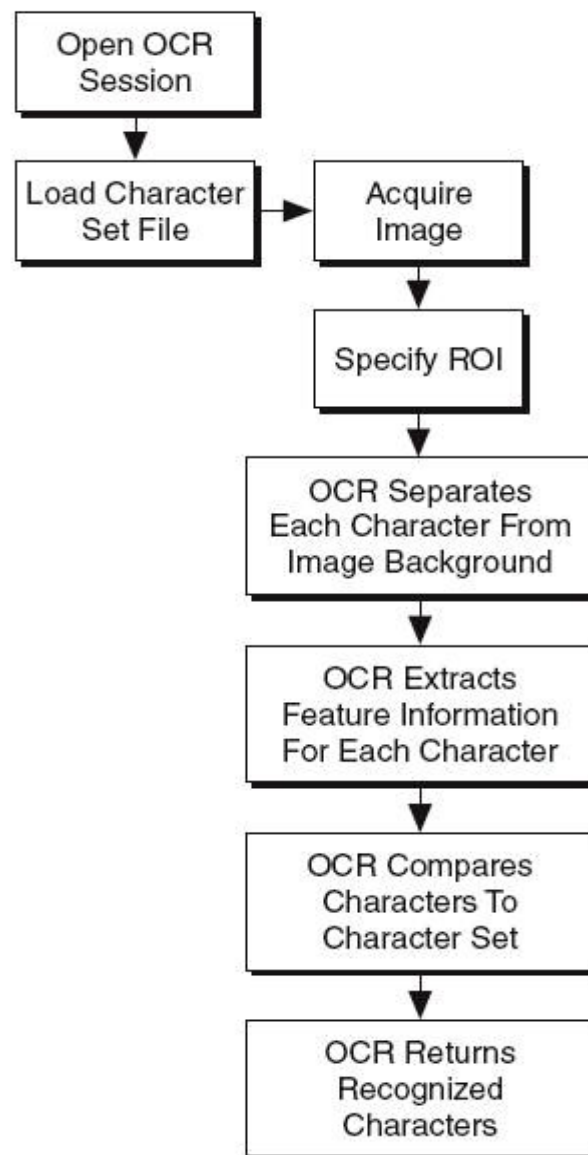


**Fig 3.12 Steps of an OCR training procedure**

The process of locating characters in an image is often referred to as character segmentation. Before we can train characters, we must set up OCR to determine the criteria that segment the characters you want to train. When we finish segmenting the characters, we'll use OCR to train the characters, storing information that enables OCR to recognize the same characters in other images. We train the OCR software by providing a character value for each of the segmented characters, creating a unique representation of each segmented character. You then save the character set to a character set file to use later in an OCR reading procedure.

### 3.4.4 Reading Characters

When we perform the reading procedure, the machine vision application we created with OCR functions segments each object in the image and compares it to characters in the character set you created during the training procedure. OCR extracts unique features from each segmented object in the image and compares each object to each character stored in the character set. OCR returns the objects that best match characters in the character set as the recognized characters.



**Fig 3.13 Steps of an OCR reading procedure**

### **3.4.5 OCR Session**

An OCR session applies to both the training and reading procedures. An OCR session prepares the software to identify a set of characters during either the training procedure or the reading procedure. A session consists of the properties we set and the character set that we train or read from a file. OCR uses session information to compare objects with trained characters to determine if they match. If we want to process an image containing characters that we stored in multiple character sets, use multiple OCR sessions simultaneously to read all the characters simultaneously. We also can merge several character sets in one session. If we choose to merge multiple character sets, train each of the character sets with the same segmentation parameters.

### **3.4.6 Region of Interest (ROI)**

The ROI applies to both the training and reading procedures. During training, the ROI is the region that contains the objects we want to train. During reading, the ROI is the region that contains the objects we want to read by comparing the objects to the character set. We can use the ROI to effectively increase the accuracy and efficiency of OCR. During training, we can use the ROI to carefully specify the region in the image that contains the objects we want to train while excluding artifacts. During reading, we can use the ROI to enclose only the objects we want to read, which reduces processing time by limiting the area OCR must analyze.

### **3.4.7 Particles, Elements, Objects, and Characters**

Particles, elements, objects, and characters apply to both the training and reading procedures. Particles are groups of connected pixels. Elements are particles that are part of an object. For example, the dots in a dot-matrix object are elements. A group of one or more elements forms an object based on the element spacing criteria. A character is a trained object.

### **3.4.8 Character Segmentation**

Character segmentation applies to both the training and reading procedures. Character segmentation refers to the process of locating and separating each character in the image from the background.

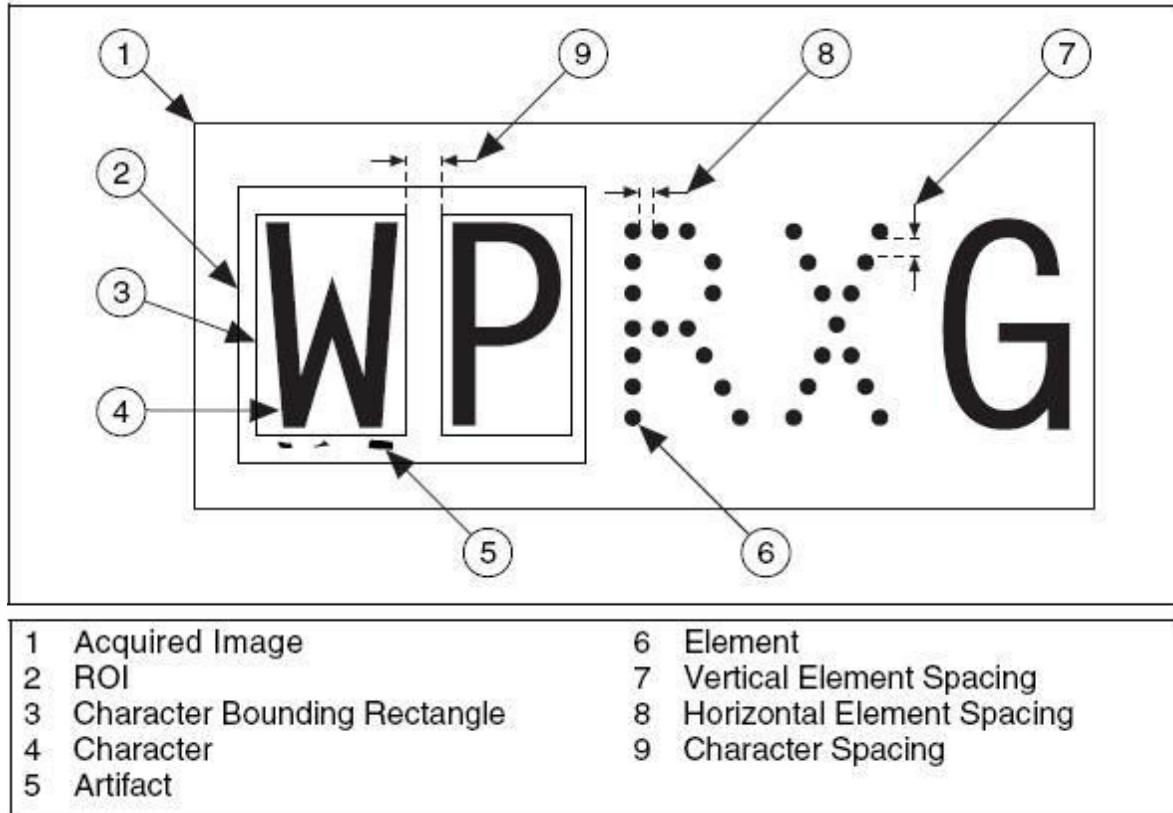


Fig 3.14 Concepts involved in Character segmentation

### 3.4.9 Thresholding

Thresholding is one of the most important concepts in the segmentation process. Thresholding is separating image pixels into foreground and background pixels based on their intensity values. Foreground pixels are those whose intensity values are within the lower and upper threshold values of the threshold range. Background pixels are those whose intensity values lie outside the lower and upper threshold values of the threshold range. OCR includes one manual method and three automatic methods of calculating the thresholding range:

**Fixed Range** is a method by which you manually set the threshold value. This method processes grayscale images quickly, but requires that lighting remain uniform across the ROI and constant from image to image. The following three automatic thresholding methods are affected by the pixel intensity of the objects in the ROI. If the objects are dark on a light background, the automatic methods calculate the high threshold value and set the low threshold value to the lower value of the threshold limits. If the objects are light on a dark background, the automatic methods

calculate the low threshold value and set the high threshold value to the upper value of the threshold limits.

□ **Uniform** is a method by which OCR calculates a single threshold value and uses that value to extract pixels from items across the entire ROI. This method is fast and is the best option when lighting remains uniform across the ROI.

□ **Linear** is a method that divides the ROI into blocks, calculates different threshold values for the blocks on the left and right side of an ROI, and linearly interpolates values for the blocks in between. This method is useful when one side of the ROI is brighter than the other and the light intensity changes uniformly across the ROI.

□ **Non linear** is a method that divides the ROI into blocks, calculates a threshold value for each block, and uses the resulting value to extract pixel data.

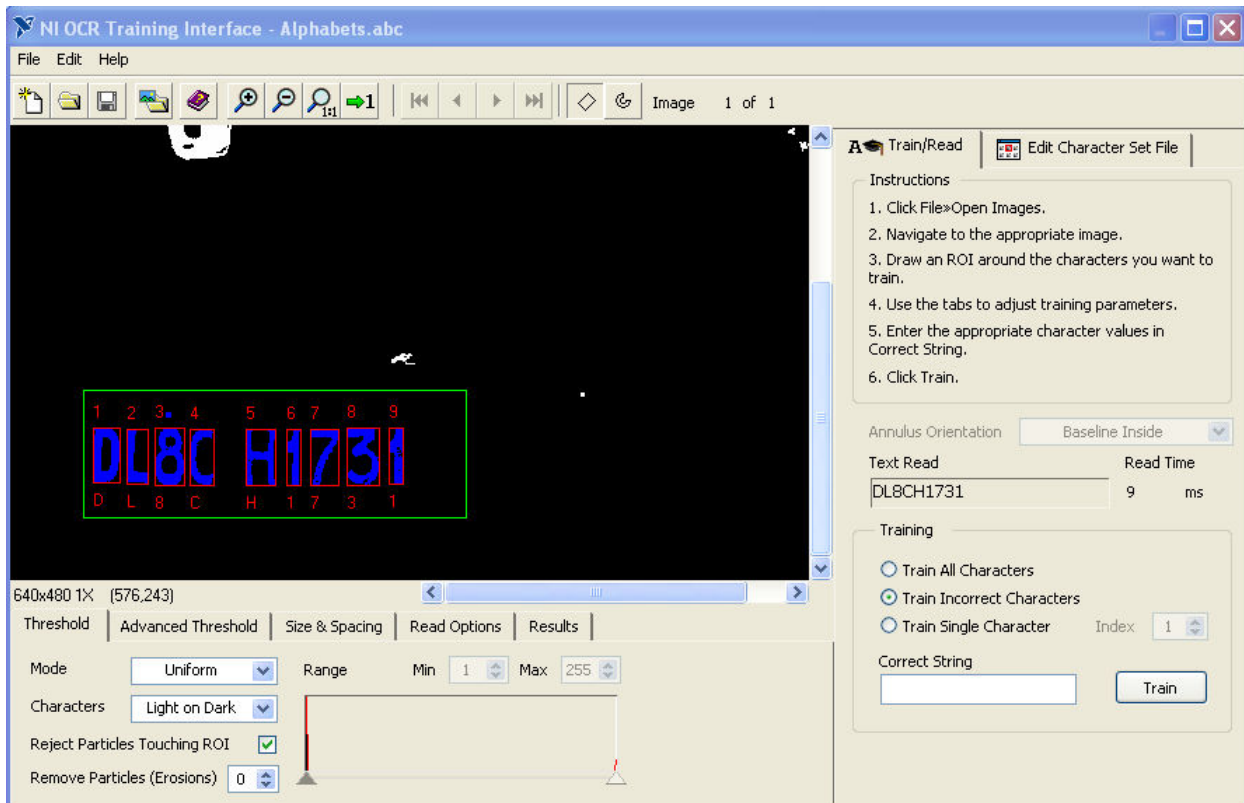
OCR includes a method by which you can improve performance during automatic thresholding, which includes the Uniform, Linear, and Non linear methods.

□ **Optimize for Speed** allows us to determine if accuracy or speed takes precedence in the threshold calculation algorithm. If speed takes precedence, enable Optimize for Speed to perform the thresholding calculation more quickly, but less accurately. If accuracy takes precedence, disable Optimize for Speed to perform the thresholding calculation more slowly, but more accurately

If we enable Optimize for Speed, we also can enable Bi modal calculation to configure OCR to calculate both the lower and upper threshold levels for images that are dominated by two pixel intensity levels.

### **3.4.10 Threshold Limits**

Threshold limits are bounds on the value of the threshold calculated by the automatic threshold calculation algorithms. For example, if the threshold limits are 10 and 240, OCR uses only intensities between 10 and 240 as the threshold value. Use the threshold limits to prevent the OCR automatic threshold algorithms from returning too low or too high values for the threshold in a noisy image or an image that contains a low population of dark or light pixels. The default range is 0 to 255.



**Fig 3.15 An OCR Training Interface**

### 3.4.11 Character Spacing

Character spacing is the horizontal distance, in pixels, between the right edge of one character bounding rectangle and the left edge of the next character bounding rectangle. If an image consists of segmented or dot-matrix characters and the spacing between two characters is less than the spacing between the elements of a character, we must use individual ROIs around each character.

### 3.4.12 Element Spacing

Element spacing consists of horizontal element spacing and vertical element spacing. Horizontal element spacing is the space between two horizontally adjacent elements. Set this value to 1 or 2 for stroke characters and 4 or 5 for dot-matrix or segmented characters. Dot-matrix or segmented characters are characters comprised of a series of small elements. Stroke characters are continuous characters in which breaks are due only to imperfections in the image. If we set the horizontal element spacing too low, we might accidentally eliminate elements of an object. If we set the horizontal element spacing too high, we might include extraneous elements in the object, resulting in a trained object that does not represent a matchable character. Vertical element



spacing is the space between two vertically adjacent elements. Use the default value, 0, to consider all elements within the vertical direction of the ROI to be part of an object. If we set vertical element spacing too high, we might include artifacts as part of an object.

### **3.4.13 Character Bounding Rectangle**

The character bounding rectangle is the smallest rectangle that completely encloses a character.

#### **(a) Auto Split**

Auto Split applies to both the training and reading procedures. Use AutoSplit when an image contains characters that are slanted. AutoSplit, which works in conjunction with the maximum character bounding rectangle width, uses an algorithm to analyze the right side of a character bounding rectangle and determine the rightmost vertical line in the object that contains the fewest number of pixels. AutoSplit moves the rightmost edge of the character bounding rectangle to that location. The default value is False.

#### **(b) Character Size**

Character size is the total number of pixels in a character. Generally, character size should be between 25 and 40 pixels. If characters are too small, training becomes difficult because of the limited data. The additional data included with large characters is not helpful in the OCR process, and the large characters can cause the reading process to become very slow.

#### **(c) Substitution Character**

Substitution character applies to the reading procedure only. OCR uses the substitution character for unrecognized characters. The substitution character is a question mark (?) by default.

#### **(d) Acceptance Level**

Acceptance level applies to the reading procedure. Acceptance level is a value that indicates how closely a read character must match a trained character to be recognized. The valid range for this value is 0 to 1000. The higher the value, the more closely you expect an object to match a trained character. The default value is 700.

### **3.4.14 Read Strategy**

Read strategy applies only to the reading procedure. Read strategy refers to the criteria OCR uses to determine if a character matches a trained character in the character set. The possible modes are Aggressive and Conservative. In Aggressive mode, the reading procedure uses fewer criteria than Conservative mode to determine if an object matches a trained character. Aggressive mode works well for most applications. In Conservative mode, the reading procedure uses extensive criteria to determine if an object matches a trained character.

### **3.4.15 Read Resolution**

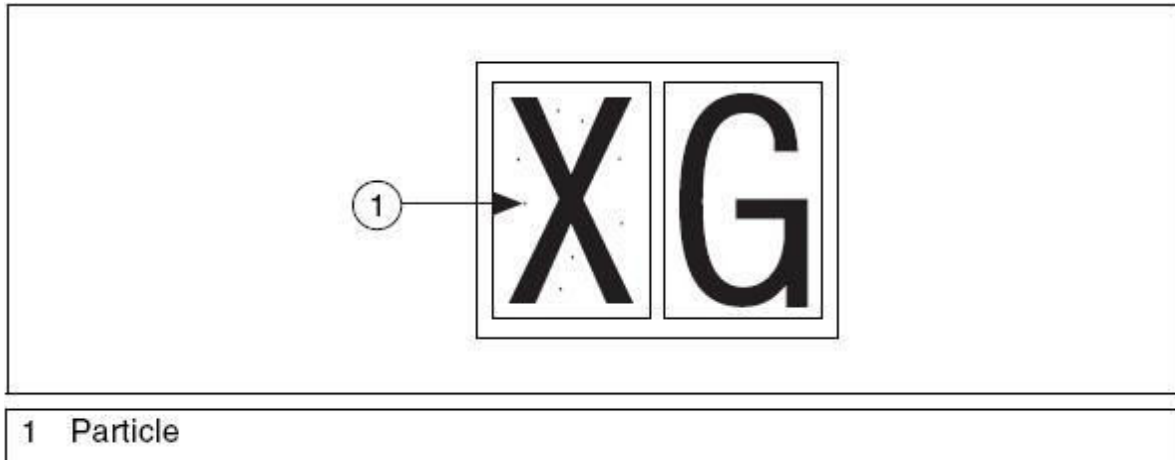
Read resolution applies to the reading procedure. When we save a character set, OCR saves a variety of information about each character in the character set. Read resolution is the level of character detail OCR uses to determine if an object matches a trained character. By default, OCR uses a low read resolution, using few details to determine if there is a match between an object and a trained character. The low read resolution enables OCR to perform the reading procedure more quickly. We can configure OCR to use a medium or high read resolution, and therefore use more details to determine if an object matches a trained character. Using a high read resolution reduces the speed at which OCR processes. The low resolution works well with most applications, but some Applications might require the higher level of detail available in medium or high resolutions.

### **3.4.16 Valid Characters**

Valid characters apply only to the reading procedure. Valid characters refer to the practice of limiting the characters that the reading procedure uses when analyzing an image. For example, if you know that the first character in an ROI should be a number, you can limit the reading procedure to comparing the first character in the ROI only to numbers in the character set. Limiting the characters that the reading procedure uses when analyzing an image increases the speed and accuracy of OCR.

### **3.4.16 Removing Small Particles**

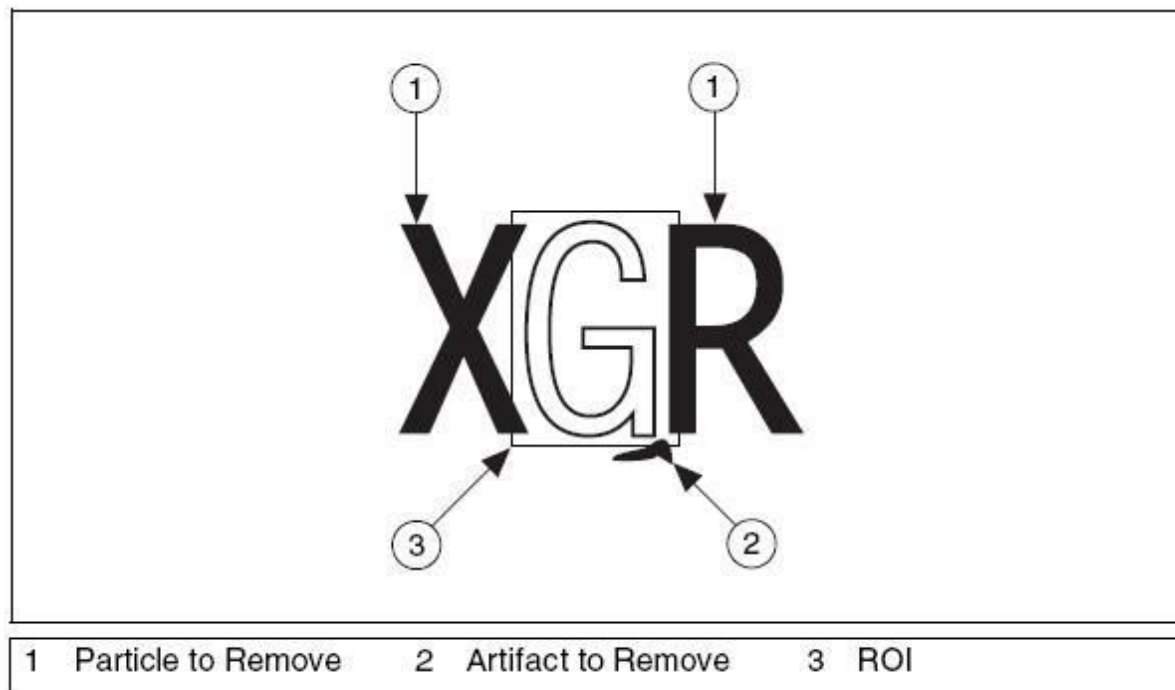
Removing small particles applies to both the training and reading procedures. The process of removing small particles involves applying a user-specified number of 3x3 erosions to the thresholded image. OCR fully restores any objects that remain after applying the erosions. For example, in Figure 3.11 if any portion of the letters X and G remains after removing small particles, OCR fully restores the X and G.



**Fig 3.16 Unwanted Particles in an image**

### 3.4.17 Removing Particles That Touch the ROI

Removing particles that touch the ROI applies to both the training and reading procedures.



**Fig 3.17 Unwanted Particles that touch ROI**

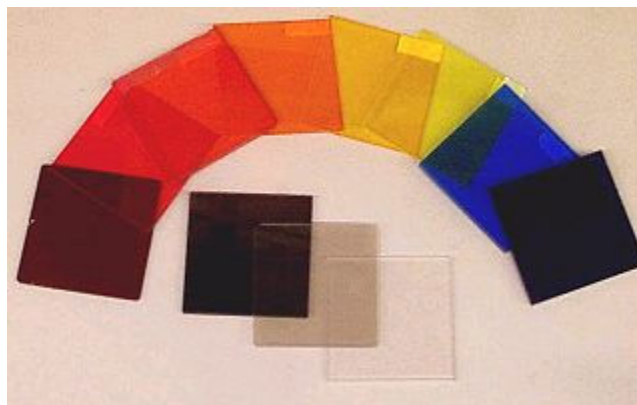
We can configure OCR to remove small particles that touch an ROI we specified. Figure 3.12 shows particles that touch the ROI and these particles can be neglected by checking the remove particles touching ROI check box in an OCR training interface.

# Chapter 4

## Camera Optics

### 4.1 Introduction to Optical Filters

Optical filters, generally, belong to one of two categories. The simplest, physically, is the **Absorptive Filter**, while the latter category, that of interference or **Dichroic Filter**, can be quite complex. Optical filters selectively transmits light having certain properties (often, a particular range of wavelengths, that is, range of color of light), while blocking the remainder. They are commonly used in photography, in many optical instruments, and to color stage lighting. In astronomy, optical filters can be used to eliminate light from the Sun or from a star much brighter than the target object, Even seeing Mercury can necessitate using optical filters from extreme northern latitudes like Scandinavia or Alaska where the sky is invariably bright whenever the planet is above the horizon due to the low angle at far elongation.



**Figure 4.1 Coloured and Neutral Density filters**

### **4.1.1 Absorptive Filters**

Absorptive filters are usually made from glass to which various inorganic or organic compounds have been added. These compounds absorb some wavelengths of light while transmitting others. The compounds can also be added to plastic (often polycarbonate or acrylic) to produce gel filters, which are lighter and cheaper than glass-based filters.

### **4.1.2 Dichroic Filter**

Alternately, dichroic filters (also called "reflective" or "thin film" or "interference" filters) can be made by coating a glass substrate with a series of optical coatings. Dichroic filters usually reflect the unwanted portion of the light and transmit the remainder.

Dichroic filters use the principle of interference. Their layers form a sequential series of reflective cavities that resonate with the desired wave lengths. Other wavelengths destructively cancel or reflect as the peaks and troughs of the waves overlap.

They can be used in devices such as the dichroic prism of a camera to separate a beam of light into different coloured components.

### **4.1.3 Monochromatic**

Monochromatic filters only allow a narrow range of wavelengths (that is, a single color) to pass.

### **4.1.4 Infrared**

Infrared (IR) or heat-absorbing filters are designed to block or reflect mid-infrared wavelengths but pass visible light. They are often used in devices with bright incandescent light bulbs (such as slide and overhead projectors) to prevent unwanted heating. There are also filters which are used in solid state video cameras to block IR due to the high sensitivity of many camera sensors to near-infrared light.

### **4.1.5 Ultraviolet**

Ultraviolet (UV) filters block ultraviolet radiation, but let visible light through. Because photographic film and digital sensors are sensitive to ultraviolet (which is abundant in skylight) but the human eye is not, such light would, if not filtered out, make photographs look different from the scene that the photographer saw. This causes images of distant mountains to appear hazy. By attaching a filter to remove ultraviolet, photographers can produce pictures that more closely resemble the scene as seen by a human eye.

### **4.1.6 Neutral density**

Neutral density (ND) filters have a constant attenuation across the range of visible wavelengths, and are used to reduce the intensity of light by reflecting or absorbing a portion of it. They are specified by the optical density (OD) of the filter, which is the negative of the common logarithm of the transmission coefficient. They are useful for making photographic exposures longer. A practical example is making a waterfall look blurry when it is photographed in bright light. Alternatively, the photographer might want to use a larger aperture (so as to limit the depth of field); adding an ND filter permits this. ND filters can be reflective (in which case they look like partially-reflective mirrors) or absorptive (appearing grey or black).

### **4.1.7 Longpass**

A longpass (LP) Filter is an optical interference or coloured glass filter that attenuates shorter wavelengths and transmits (passes) longer wavelengths over the active range of the target spectrum (ultraviolet, visible, or infrared). Longpass filters, which can have a very sharp slope (referred to as edge filters), are described by the cut-on wavelength at 50 percent of peak transmission. In fluorescence microscopy, longpass filters are frequently utilized in dichroic mirrors and barrier (emission) filters. Use of the older term of highpass to describe longpass filters is now discouraged because it more accurately refers to frequency rather than wavelength.

### **4.1.8 Shortpass**

A shortpass (SP) Filter is an optical interference or coloured glass filter that attenuates longer wavelengths and transmits (passes) shorter wavelengths over the active range of the target spectrum (usually the ultraviolet and visible region). In fluorescence microscopy, shortpass filters are frequently employed in dichromatic mirrors and excitation filters.

### **4.1.9 Bandpass**

Combining an LP filter and an SP filter produces a Bandpass (BP) filter. These filters usually have lower transmittance values than SP and LP filters, and block all wavelengths outside a selected interval, which can be wide or narrow, depending on the number of layers of the filter.

### **4.1.10 Metal Mesh Filters**

Filters for sub-millimeter and near infrared wavelengths in astronomy are metal mesh grids that are stacked together to form LP, BP, and SP filters for these wavelengths.

### **4.1.11 Polarizer**

Another kind of optical filter is a polarizer or polarization filter, which blocks or transmits light according to its polarization. They are often made of materials such as Polaroid and are used for sunglasses and photography. Reflections, especially from water and wet road surfaces, are partially polarized, and polarized sunglasses will block some of this reflected light, allowing an angler to better view below the surface of the water and better vision for a driver. Light from a clear blue sky is also polarized, and adjustable filters are used in colour photography to darken the appearance of the sky without introducing colours to other objects, and in both colour and black-and-white photography to control specular reflections from objects and water.

Polarized filters are also used to view certain types of stereograms, so that each eye will see a distinct image from a single source.

There are two types of **polarizing filters**, **polarizers** for short: **linear** and **circular**. Most digital cameras use circular polarizers due to the use of beamsplitters. Non-DSLR digital cameras in general do not use beamsplitters, and, as a result, can use both (*i.e.*, linear or circular). Polarizers can greatly enhance outdoor images by increasing contrast and color saturation, eliminating reflections from glass and other non-metallic surfaces, and darkening blue sky.

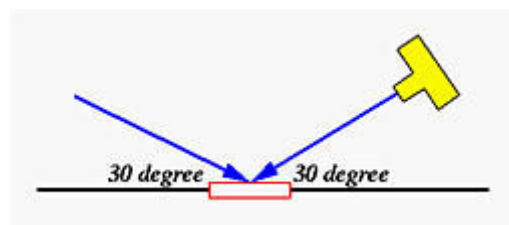


**Figure 4.2 Polarizer Lens**

A polarizer has two rings. The bottom ring screws on to the front thread of a lens, and the top ring is rotatable. As the top ring rotates, one can see the effect through the LCD monitor.

## **Eliminating Reflections**

For maximum polarization, the incoming light must have an incident angle of 30 degree, and the camera is aimed at the subject at an angle of 30 degree from the other side (see image below).



**Figure 4.3 Light Reflection Angle**



The following images demonstrate a polarization effect. Without using a polarizer (left image), the surfaces of the windows reflect the color of the sky and surrounding environment. When a polarizer is used and rotated to yield the maximum effect, reflection of the top level windows disappears completely, while reflection of the lower level windows is reduced.



**Figure 4.4a Without a polarizer**



**Figure 4.4b With a polarizer and max. effect**

The following shows a more dramatic example. The left image was taken without the use of a polarizer and the right one was taken with a polarizer. As you can see from these results, reflection from water washes out all colors, making the image very foggy.



**Figure 4.5a Without a polarizer**



**Figure 4.5b With a polarizer and max. effect**

## 4.2 Introduction to Auto Iris lens

The Auto Iris Lens is a type of lens that can be electronically controlled. This type of lens allows maintaining the lighting level. It is applicable where the light condition varies continuously.

Auto iris lens allows the camera to adjust to get the best picture in the given lighting conditions.

There are two types of Auto iris lenses:

- 1) Fixed Auto Iris lenses and
- 2) Vari focal Auto Iris lens.

Auto iris lenses are best lens to use in any application as they are flexible and can adjust to changing light levels. Auto iris lenses are used to produce consistent video signals from cameras that view scenes in any varying light levels. Auto iris lenses combines an electronically controlled iris, which allows the lens to maintain one light level making them ideal for outdoor use or any other applications.

### 4.2.1 Day/Night Auto-Iris Varifocal Lens

The Day/Night Cameras which function as color cameras in the daytime and high-sensitive monochrome cameras in the night now come onstage as the surveillance systems expanded in their applications from the daytime to the night. However, ordinary lenses do not reproduce crisp images under IR Illuminations as sharp as the images under visible lights during daytime. That is due to chromatic aberration that causes the shift of focusing points under visible lights and IR Rays. This Day/Night lens is chromatic aberration corrected extensively in a wide range from visible lights to near IR Rays and is designed to reduce the focus shift to the unrecognizable level. It displays outstanding optical quality and performance in the range from visible lights to near IR Rays.



**Figure 4.6 Day/Night Auto-Iris Varifocal Lens**

# Chapter 5

## System Development and Simulation

### 5.1 Introduction to LabVIEW

LabVIEW or Laboratory Virtual Instrument Engineering Workbench is graphical programming language software used for data acquisition and instrument control. The programs that take a long time using conventional programming languages can be completed in hours using LabVIEW. The LabVIEW programs are called virtual instruments (*VI*s) because their appearance and operation imitate actual instruments. A VI has two main parts

**Front panel** – It is the interactive user interface of a VI. The front panel can contain knobs, push buttons etc. which are the interactive input controls and graphs, LED's etc. which are indicators. Controls simulate instrument input devices and supply data to block diagram of the VI. Indicators simulate output devices and display the data that block diagram generates. The front panel objects have corresponding terminal on the block diagram

**Block diagram** – It is the VI's source code, constructed in LabVIEW's graphical programming language, G. The block diagram is the actual executable program. The components of the block diagram are lower level VIs, built in functions, constants and program execution control structures. Wires are used to connect the objects together to indicate the flow of data between them. The data that we enter into the front panel controls enter the block diagram through these terminals and after execution the output data flow to indicator terminals where they exit the block diagram, re-enter the front panel and appear in front panel indicators giving us final results. Various steps involved in Vision Assistant to create LabVIEW VI are:

1. Click **Tools Create LabVIEW VI**.
2. Browse path for creating a new file in VI.
3. Click **Next**.
4. If we want to create LabVIEW of current script click on **Current script** option or else click on **Script file** option and give the path for the file in the field given for browsing. Click **Next**.
5. Select image source (Image File).
6. Click **Finish**.

## **5.2 Introduction to IMAQ Vision**

IMAQ Vision for LabVIEW is a library of LabVIEW VIs that you can use to develop machine vision and scientific imaging applications. National Instruments also offers IMAQ Vision for Measurement Studio, which includes the same imaging functions for LabWindows™/CVI™ and other C development environments and includes ActiveX controls for Visual Basic. Vision Assistant, another software product from NI, allows you to prototype your application strategy quickly without having to do any programming. Additionally, NI offers Vision Builder for Automated Inspection, configurable machine vision software that you can use to prototype, benchmark, and deploy applications.

### **5.2.1 Acquiring Image**

NI-IMAQ for USB Cameras provides the ability to use USB cameras that have Direct Show Filters with the NI Vision Assistant or LabVIEW and IMAQ Vision. The cameras may operate at various resolutions and frame rates, depending on camera capabilities. NI-IMAQ for USB Cameras can acquire and set properties using the camera manufacturer driver and Direct Show functions. For NI-IMAQ for USB Cameras to use the camera successfully, the camera manufacturer driver needs to be correctly installed and recognized by the operating system. Refer to your camera documentation for instructions about how to correctly install the manufacturer driver.

### **Building Applications with NI-IMAQ for USB Cameras**

The NI-IMAQ for USB Cameras Virtual Instrument (VI) Library is a series of LabVIEW VIs for using LabVIEW with your USB camera and is included with the NI-IMAQ for USB Cameras software. IMAQ Vision for LabVIEW is an image processing and analysis library consisting of more than 400 VIs. IMAQ Vision for LabVIEW is required to use the NI-IMAQ for USB Cameras VI library. The following sections describe the basic concepts and information necessary to build an application with NI-IMAQ for USB Cameras.

## Architecture

Figure 5.1 shows a block diagram of the NI-IMAQ for USB Cameras architecture.

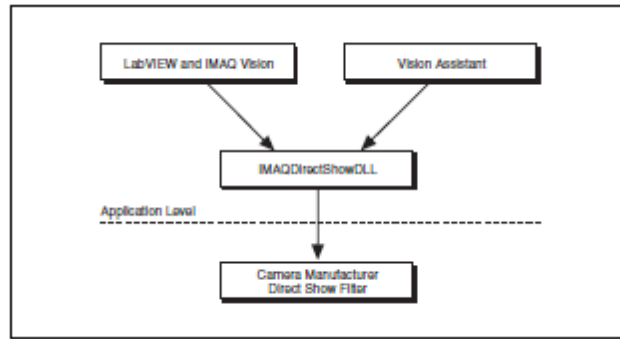


Figure 5.1 NI-IMAQ for USB Cameras Architecture

## NI-IMAQ for USB Cameras VI Location

To access the NI-IMAQ for USB Cameras VIs from within LabVIEW, select **NI Measurements»Vision»IMAQ USB** from the Functions palette of your block diagram, as shown in Figure 5.2.

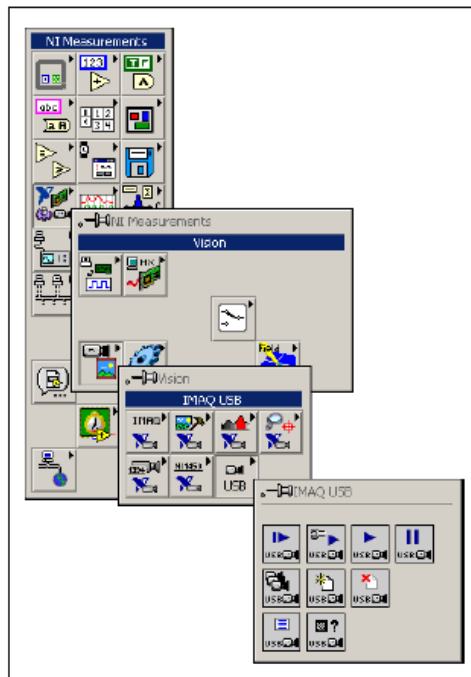


Figure 5.2. NI-IMAQ for USB Cameras Palette Location

## Buffer Management

The IMAQ Create and IMAQ Dispose VIs manage image buffers in LabVIEW. IMAQ Create, shown in Figure 3, allocates an image buffer. **Image Name** is a label for the buffer created. Each buffer must have a unique name. **Image Type** specifies the type of image being created. Use **Grayscale (U8)** for 8-bit monochrome images, **Grayscale (I16)** for 16-bit monochrome images, and **RGB (U32)** for RGB color images. **Note** If **Image Type** is set to a value incompatible with the current video mode, the **Image Type** value automatically changes to a compatible value. **New Image** contains pointer information to the buffer, which is initially empty. When you wire **New Image** to the **Image** terminal of an image acquisition VI, the image acquisition VI allocates the correct amount of memory for the acquisition. If you are going to process the image, you may also need to wire a value to **Border Size**. **Border Size** is the width, in pixels, of a border created around an image. Some image processing functions, such as labeling or morphology, require a border.

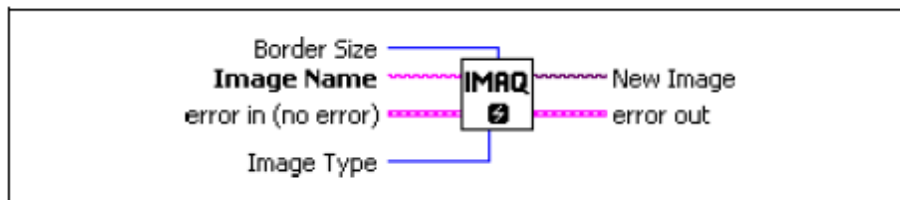


Figure 5.3. IMAQ Create

IMAQ Dispose, shown in Figure 4, frees the memory allocated for the image buffer. Call this VI only after the image is no longer required for processing.



Figure 5.4. IMAQ Dispose

## Acquisition Types

Two types of image acquisitions are available in LabVIEW—snap and grab. The following sections describe and give examples for each acquisition type. NI-IMAQ for USB Cameras does not support simultaneous acquisitions from more than one camera.

## 1) Snap

A *snap* acquires a single image into a memory buffer. Use this acquisition mode to acquire a single frame to a buffer. When you invoke a snap, it initializes the device and acquires the next incoming video frame to a buffer. Use a snap for low-speed or single-capture applications. Use IMAQ USB Snap to perform a snap acquisition in LabVIEW. Figure 5.5 shows a simplified block diagram for using IMAQ USB Snap.

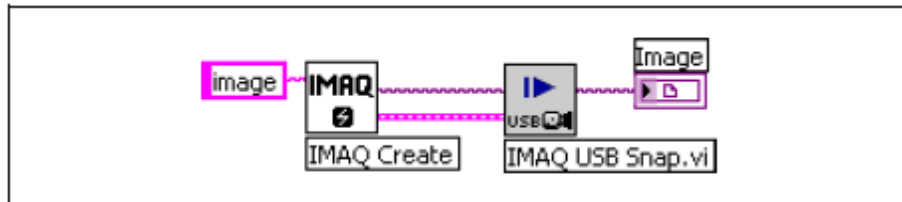


Figure 5.5 Acquiring an Image Using IMAQ USB Snap

## 2) Grab

A *grab* is a continuous, high-speed acquisition of data to a single buffer in host memory. This function performs an acquisition that loops continually on one buffer. You can get a copy of the acquisition buffer by grabbing a copy to a LabVIEW image buffer. You must use two VIs— IMAQ USB Grab Setup and IMAQ USB Grab Acquire—for a grab acquisition in LabVIEW. Call IMAQ USB Grab Setup once to initialize the acquisition and start capturing the image to an internal software buffer. You can call IMAQ USB Grab Acquire multiple times to copy the image currently stored in the internal buffer to a LabVIEW image buffer. After the program finishes copying images, call IMAQ USB Close once to shut down the acquisition. Figure 6 shows a simplified block diagram for using IMAQ USB Grab Setup and IMAQ USB Grab Acquire.

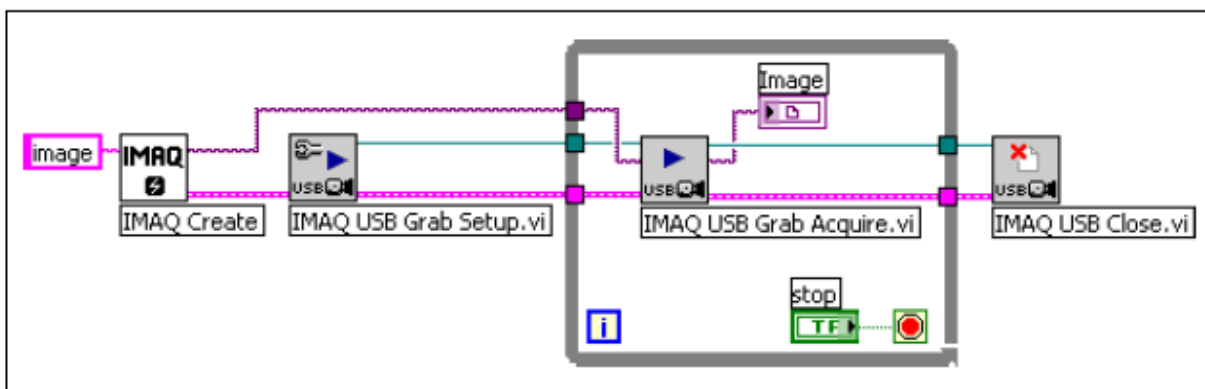


Figure5.6. Acquiring Images Using IMAQ USB Grab

## 5.2.2 IMAQ Vision Functions Palettes

### Vision Utilities



Vision Utilities functions allow you to manipulate and display images in IMAQ Vision.



- **Image Management**—A group of VIs that manage images. Use these VIs to create and dispose images, set and read attributes of an image (such as its size and offset), and copy one image to another. You also can use some of the advanced VIs to define the border region of an image and access the pointer to the image data.



- **Files**—A group of VIs that read images from files, write images to files in different file formats, and get information about the image contained in a file.



- **External Display**—A group of VIs that control the display of images in external image windows. Use these VIs to do the following:
  - Get and set window attributes, such as size, position, and zoom factor
  - Assign color palettes to image windows
  - Set up and use image browsers
  - Set up and use different drawing tools to interactively select ROIs on image windows
  - Detect draw events
  - Retrieve information about ROIs drawn on the image window





**Note** If you have LabVIEW 7.0 or later, you also can use the Image Display control available from the Vision control palette.



- **Region of Interest**—A group of VIs that manage ROIs. Use these VIs to programmatically define ROIs and convert ROIs to and from image masks.



**Note** If you have LabVIEW 7.0 or later, you can use the property node and invoke node of the Image Display control to perform many of these ROI tasks.



- **Image Manipulation**—A group of VIs that modify the spatial content of images. Use these VIs to resample an image, extract parts of an image, and rotate, shift, and unwrap images. This subpalette also contains VIs that copy images to and from the clipboard.



- **Pixel Manipulation**—A group of VIs that read and modify individual pixels in an image. Use these VIs to read and set pixel values in an image or along a row or column in an image, fill the pixels in an image with a particular value, and convert an image to and from a 2D LabVIEW array.



- **Overlay**—A group of VIs that overlay graphics on an *image display environment* without altering the pixel values of the image. Use these VIs to overlay the results of your inspection application onto the images you inspect.



- **Calibration**—A group of VIs that spatially calibrate an image to take accurate, real-world measurements regardless of camera perspective or lens distortion. Use these VIs to set a simple calibration or to let IMAQ Vision automatically learn the calibration data from a grid image. Then use the VIs to convert pixel coordinates to real-world coordinates for simple measurements.



- **Color Utilities**—A group of VIs that access data from color images. Use these VIs to extract different color planes from an image, replace the planes of a color image with new data, convert a color image to and from a 2D array, read and set pixel values in a color image, and convert pixel values from one color space to another.



- **IMAQ RT**—A group of VIs that provide functionality for using NI-IMAQ and IMAQ Vision with LabVIEW Real-Time (RT). Use these VIs to display images to Video Out on your RT system, to control the compression setting for sending images over the network, and to time bound your processing VIs on a LabVIEW RT system.

## Image Processing



Use the Image Processing functions to analyze, filter, and process images in IMAQ Vision.



- **Processing**—A group of VIs that process grayscale and binary images. Use these VIs to convert a grayscale image into a binary image using different thresholding techniques. You also can use these VIs to transform images using predefined or custom lookup tables, change the contrast information in the image, and invert the values in an image.



- **Filters**—A group of VIs that filter an image to enhance the information in the image. Use these VIs to smooth an image, remove noise, and highlight or enhance edges in the image. You can use a predefined convolution kernel or create custom convolution kernels.



- **Morphology**—A group of VIs that perform morphological operations on an image. Some of these VIs perform basic morphological operations, such as dilation and erosion, on grayscale and binary images. Other VIs improve the quality of binary images by filling holes in particles, removing particles that touch the image border, removing small particles, and removing unwanted particles based on different shape characteristics of the particle. Another set of VIs in this subpalette separate touching particles, find the skeleton of particles, and detect circular particles.



- **Analysis**—A group of VIs that analyze the content of grayscale and binary images. Use these VIs to compute the *histogram* information and grayscale statistics of an image, retrieve pixel information and statistics along any one-dimensional profile in an image, and detect and measure particles in binary images.



- **Color Processing**—A group of VIs that analyze and process color images. Use these VIs to compute the histogram of color images; apply lookup tables to color images; change the brightness, contrast, and gamma information associated with a color image; and threshold a color image. Some of these VIs also compare the color information in different images or different regions in an image using a color matching process.



- **Operators**—A group of VIs that perform basic arithmetic and logical operations on images. Use some of these VIs to add, subtract, multiply, and divide an image with other images or constants. Use other VIs in this subpalette to apply logical operations—such as AND/NAND, OR/NOR, XOR/XNOR—and make pixel comparisons between an image and other images or a constant. In addition, one VI in this

subpalette allows you to select regions in an image to process using a masking operation.



- **Frequency Domain**—A group of VIs that analyze and process images in the frequency domain. Use these VIs to convert an image from the spatial domain to the frequency domain using a two-dimensional Fast Fourier Transform (FFT) and convert from the frequency domain to the spatial domain using the inverse FFT. These VIs also extract the magnitude, phase, real, and imaginary planes of the complex image. In addition, these VIs allow you to convert complex images into complex 2D arrays and back. Also in this subpalette are VIs that perform basic arithmetic operations—such as addition, subtraction, multiplication and division—between a complex image and other images or a constant. Lastly, some of these VIs allow you to filter images in the frequency domain.

## Machine Vision



The IMAQ Machine Vision VIs are high-level VIs that simplify common machine vision tasks.



- **Select Region of Interest**—A group of VIs that allow you to select a region of interest tool, draw specific regions of interest in the image window, and return information about regions with very little programming.



- **Coordinate System**—A group of VIs that find a coordinate system associated with an object in an image. Use these VIs to find the coordinate system using either edge detection or pattern matching. You can then use this coordinate system to take measurements from other Machine Vision VIs.



- **Count and Measure Objects**—A VI that thresholds an image to isolate objects from the background and then finds and measures characteristics of the objects. This VI also can ignore unwanted objects in the image when making measurements.



- **Measure Intensities**—A group of VIs that measure the intensity of a pixel at a point or the statistics of pixel intensities along a line or rectangular region in an image.



- **Measure Distances**—A group of VIs that measure distances, such as the minimum and maximum horizontal distance between two vertically oriented edges or the minimum and maximum vertical distance between two horizontally oriented edges.



- **Locate Edges**—A group of VIs that locate vertical, horizontal, and circular edges.



- Find Patterns—A VI that learns and searches for a pattern in an image.
- Searching and Matching—A group of VIs that create and search for patterns in grayscale and color images. This subpalette also contains a VI to search for objects with predefined shapes in binary images.



- Caliper—A group of VIs that find edges along different profiles in the image. Use these VIs to find edges along a line, a set of parallel lines defined inside a rectangular region (rake), a set of parallel concentric lines defined inside an annular region (concentric rake), or a set of radial lines defined inside an annular region (spoke). You also can use these VIs to find edge pairs in the image that satisfy certain criteria.



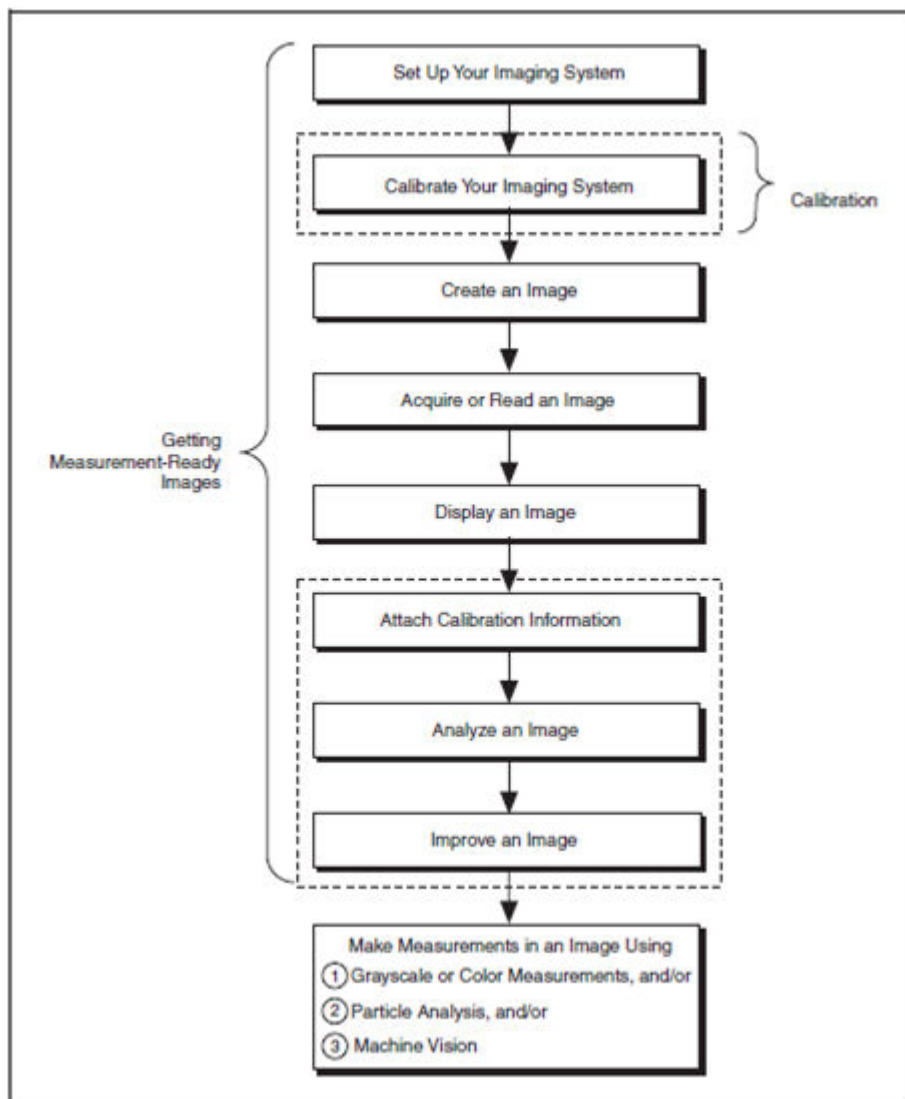
- Analytic Geometry—A group of VIs that perform analytic geometry computations on a set of points in an image. Use these VIs to fit lines, circles, and ellipses to a set of points in the image; compute the area of a polygon represented by a set of points; measure distances between points; and find angles between lines represented by points. VIs in this subpalette also perform computations, such as finding the intersection point of two lines and finding the line bisecting the angle formed by two lines.



- Instrument Readers—A group of VIs that accelerate the development of applications that require reading from seven-segment displays, meters or gauges, or one-dimensional barcodes.

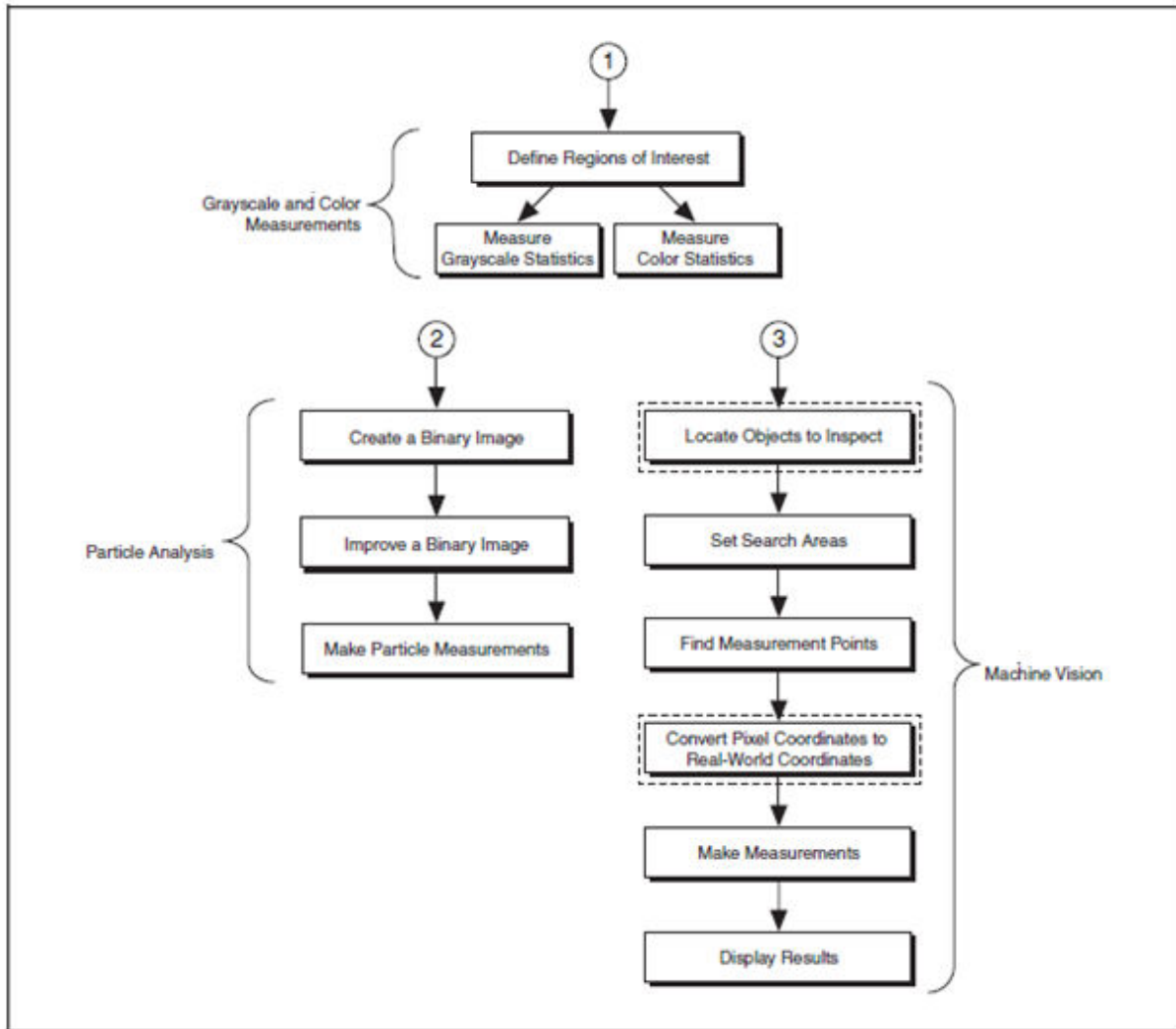
### 5.2.3 Creating IMAQ Vision Applications

Figures 5.7 illustrate the steps for creating an application with IMAQ Vision. Figure 5.7 describes the general steps to designing an IMAQ Vision application. The last step in Figure 1-1 is expanded upon in Figure 5.8. You can use a combination of the items in the last step to create your IMAQ Vision application.



**Figure 5.7 General Steps to Designing a Vision Application**





**Figure 5.8. Inspection Steps for Building a Vision Application**

### 5.3 Code Development

Once the script is converted in to LabVIEW code it will look like the image shown in figure 5.1 below. The LabVIEW code generated by our script will be performing the following operations on an image.

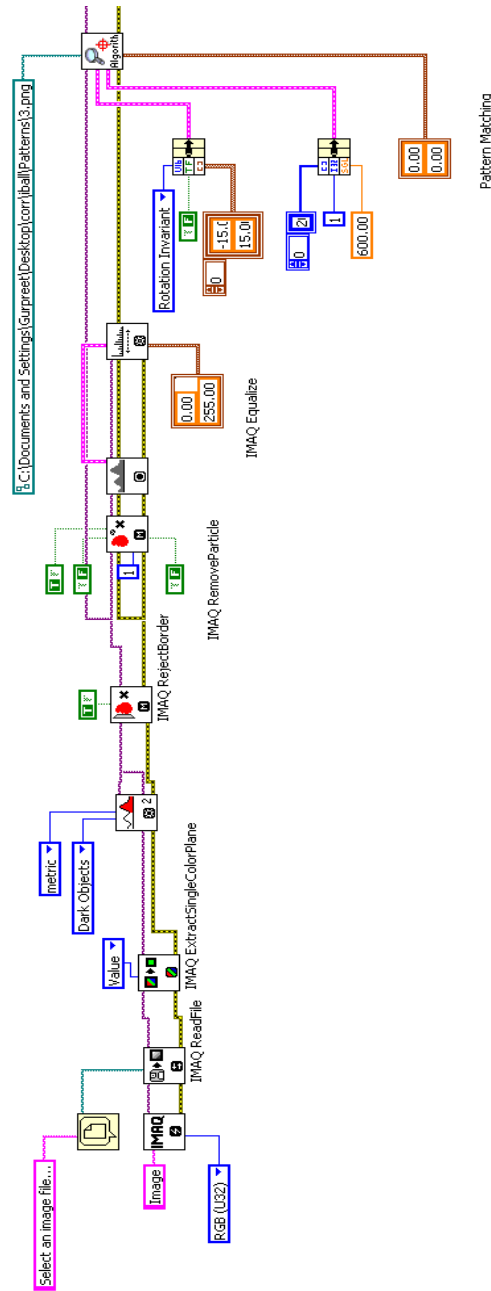
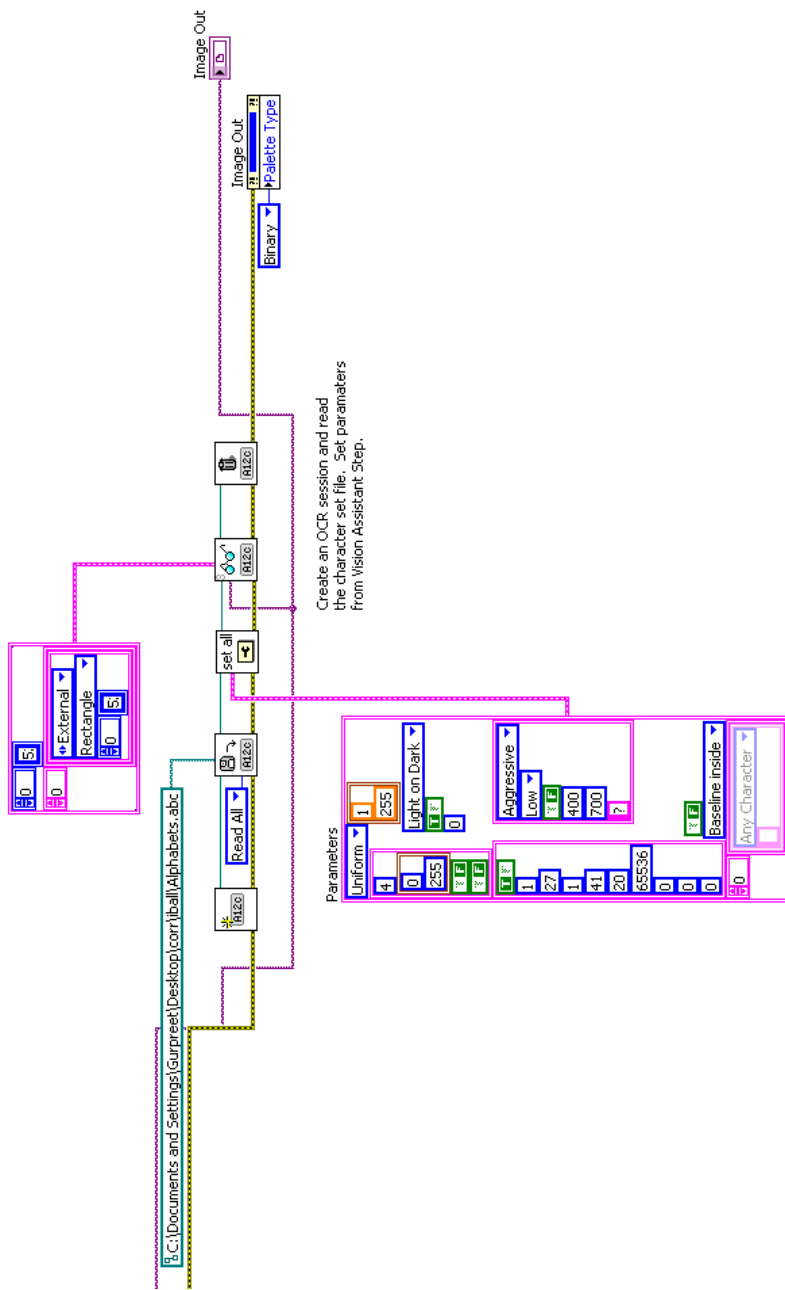


Fig 5.9 A part of LabVIEW block diagram for image acquisition filtering and pattern matching



**Fig 5.10 A part of LabVIEW block diagram for Optical Character Recognition**



# Chapter 6

## Problem Formulation & Proposed Solution

### 6.1 Problem Definition

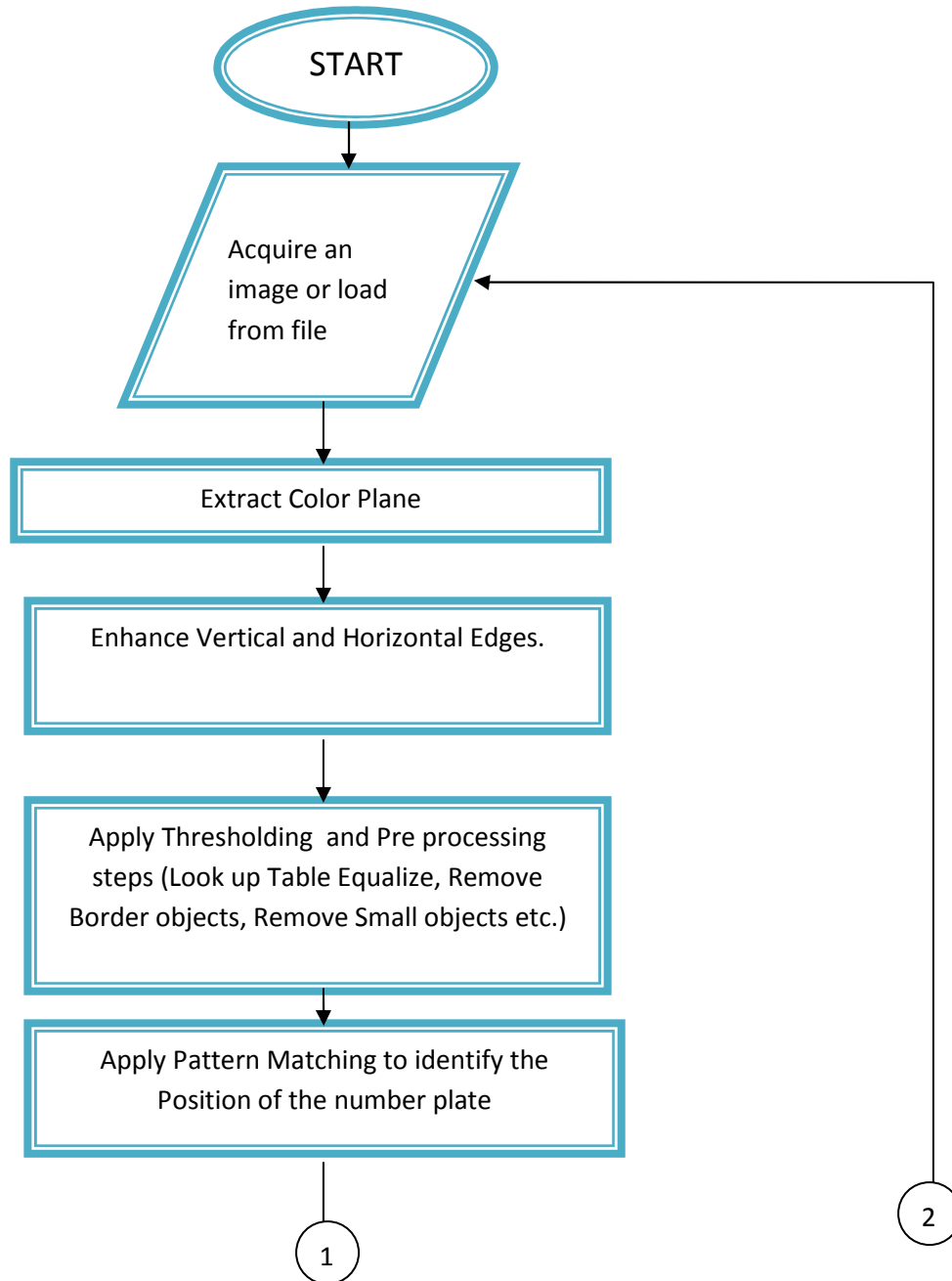
Traffic problems are significant in a developing or developed country. Massive integration of information technologies into all aspects of modern life caused demand for processing vehicles as conceptual resources in information systems. Because a standalone information system without any data has no sense, there was also a need to transform information about vehicles between the reality and information systems. This can be achieved by a human agent, or by special intelligent equipment which is able to recognize vehicles by their number plates in a real environment and reflect it into conceptual resources. Because of this, various recognition techniques have been developed and number plate recognition systems are today used in various traffic and security applications, such as parking, access and border control, or tracking of stolen cars. Till now, all the NPR systems have been developed using neural networks and MATLAB. This work proposes to implement the system using LabVIEW and Vision Assistant to make the system faster and more efficient.

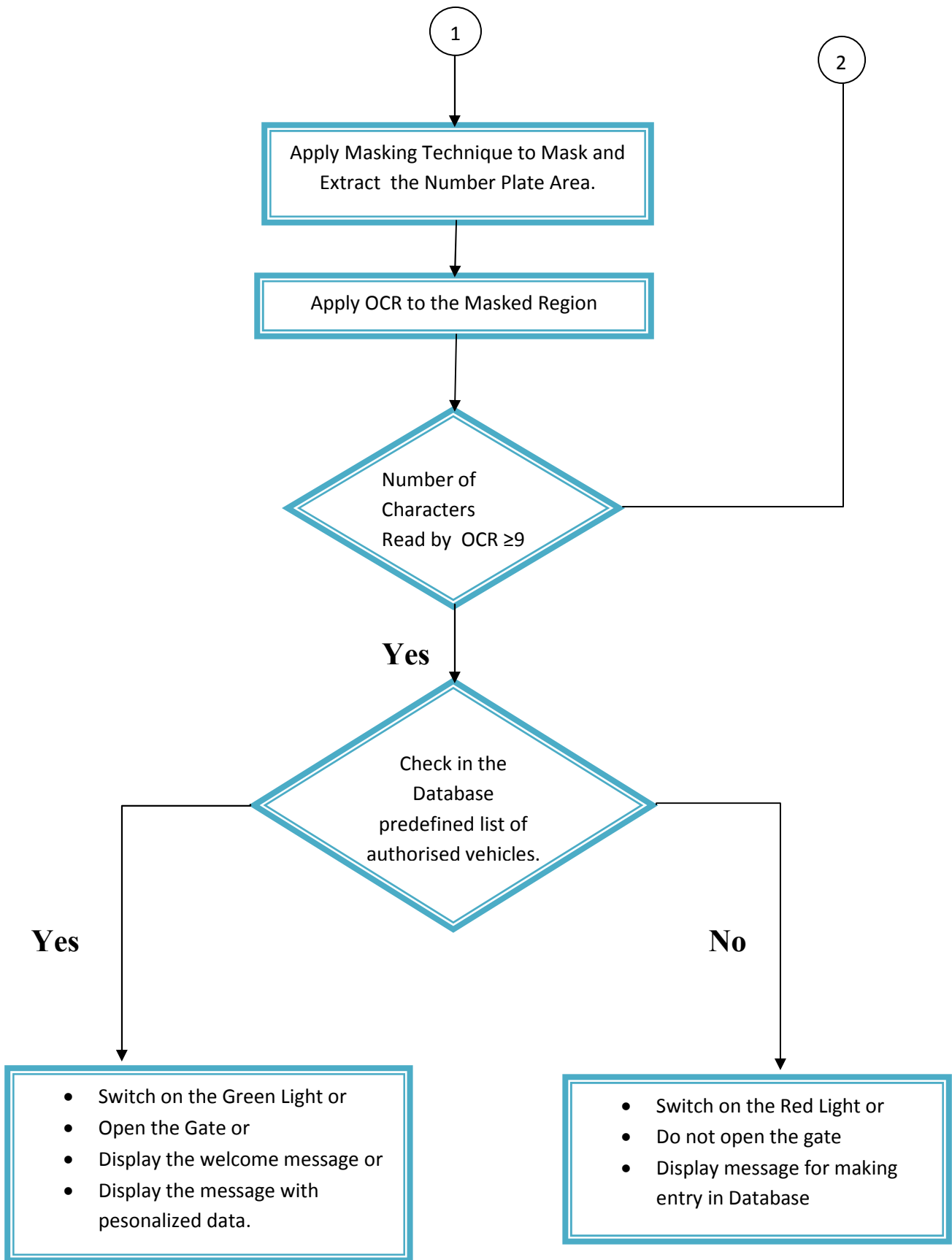
### 6.2 Proposed Solution

When the vehicle approaches the secured area, the NPR unit senses the car and activates the illumination (invisible infra-red in most cases) . The LPR unit takes the pictures from either the front or rear plates from the LPR camera. The image of the vehicle contains the number plate. The NPR unit feeds the input image to the system. The system first extracts the appropriate plane. Then apply filters to enhance the vertical and horizontal edges of the image should result in a great deal of edges in the area on number plate due to the characters. Then detect the number plate position using Pattern Matching Algorithm. Then the region with highest probability of number plate is masked and extracted. Then the resulting region of interest is scanned for characters and numerals in an OCR session. The output of OCR is saved in a database and then for each iteration the result is checked if it qualifies to contain all the digits in number plate. Whenever the results meet the conditions specified, then , Checks if the vehicle appears on a

predefined list of authorized vehicles, If found, it signals to open the gate by activating its relay. The unit can also switch on a green "go-ahead" light or red "stop" light. The unit can also display a welcome message or a message with personalized data. The authorized vehicle enters into the secured area. After passing the gate its detector closes the gate. Now the system waits for the next vehicle to approach the secured area.

. The sequence of the steps followed to develop the system is shown in Figure 4.1 below.





### 6.3 Problems Encountered

A bottleneck of any image processing project has been the camera. The now used camera is sensitive to vibrations and fast changing targets since it has quite shutter times. The optics used do not include auto-focus which would be useful. A good auto-focus makes large focus makes large focus depth unnecessary, which allows for a larger aperture that shorten the shutter time. A lower focus depth also reducing the background details, which might disturb the algorithm. A good feature reducing the effect of vibrations would be an image stabilizer. An infra-red camera along with an infrared light source would give the best performance and wheather independence would then give a good reflection in the number plate's reflective background.

For real time application, the system requires a video camera (frame grabber) with higher resolution which acquires the images of vehicles from rear or front view. But for the present work, due to unavailability of the required hardware, we have used 640x480 resolution USB Webcamera.

Due to the high cost of components (like IEEE 1394 Camera, Auto iris control Lens, Frame grabber card etc.) and limited amount of time we have, a set of constraints have been placed on our system to make the project more manageable, they are as follows:

1. In image processing the systems are greatly affected by lightening conditions, our system is also sensitive to lightening conditions and lightening conditions are hard to kept constant while you are working in the middle of a busy road.
2. The proposed system is sensitive also to the angle at which images are being taken. For better efficiencies the image must be taken in a way so that vehicle number plate comes in parallel with the Camera.
3. Image of the vehicle taken from fixed distance. Also the resolution of images must be kept above 1200x1600 for better results.
4. Vehicle is stationary when the image was taken.

5. There is no standard size of Indian number plates no standard of font style or size either which makes it quiet difficult for the software to recognize the alphanumeric characters because training the software for a specific size, shape and style is easy as compared to different. We need to give extensive training to the software in order to make it able to recognize the numbers in different cities and states as all the states have different first two characters in the vehicle number plates.


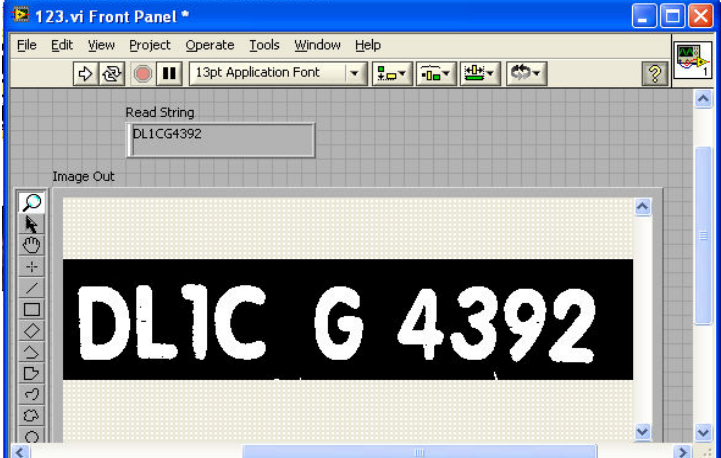

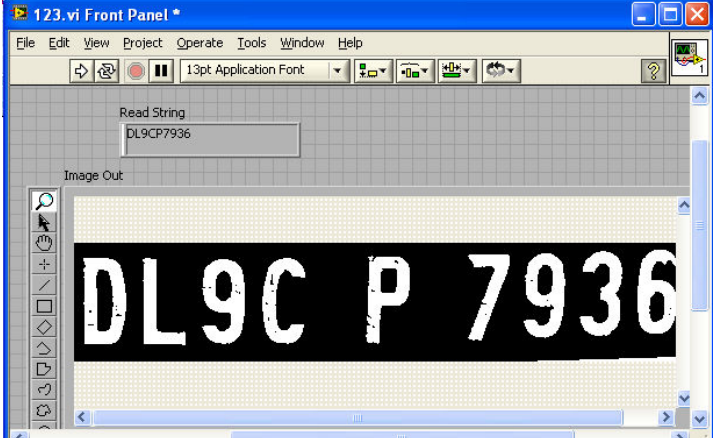


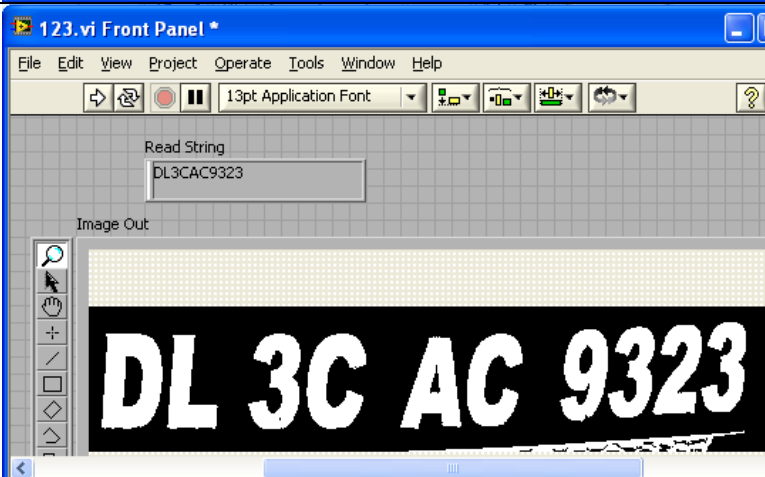
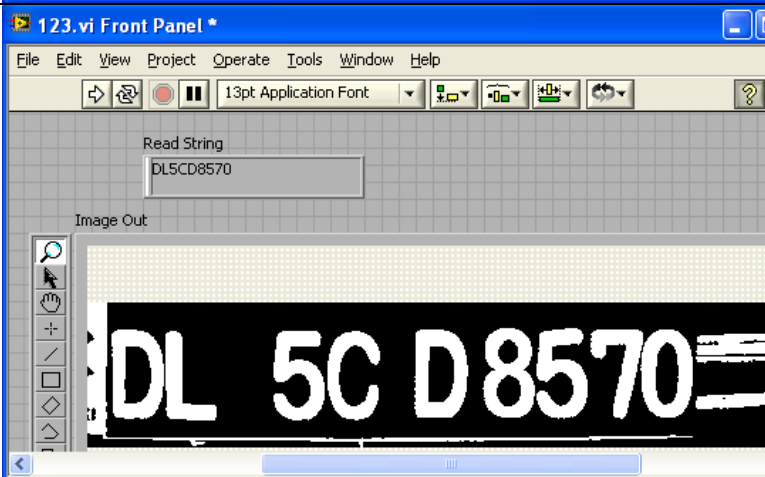
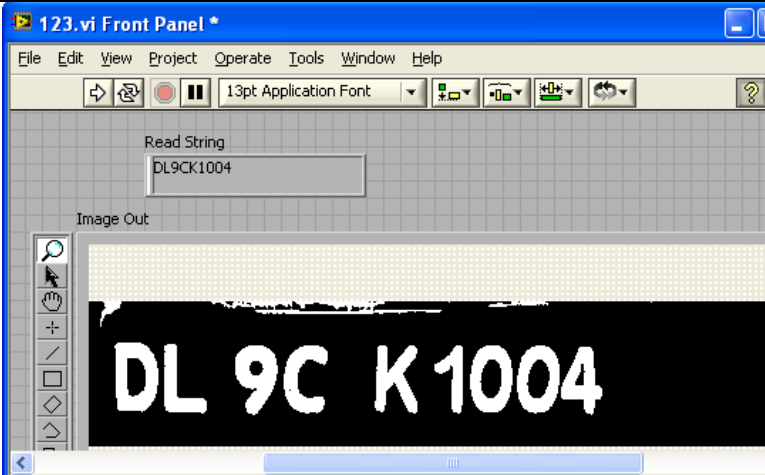
**Figure 6.1: Standard Indian Number Plate**

# Chapter 7

## Results

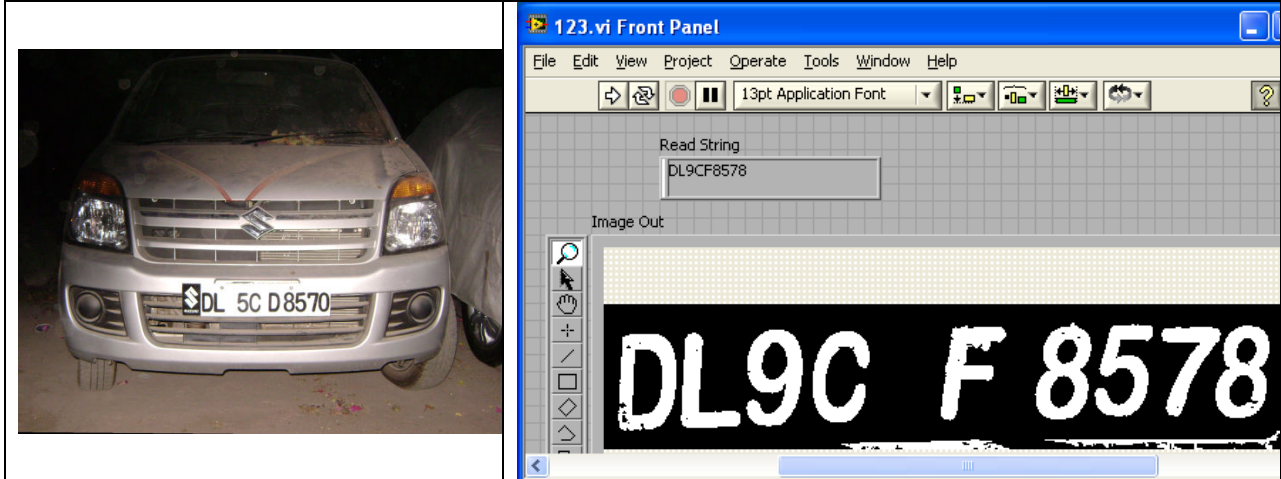
Images

Original Images	Output of Number Plate Recognition System
	
	









The purpose of this thesis was to develop a real time application which recognizes number plates from cars at a gate, for example at the entrance of a parking area or a border crossing. The system, based on regular PC with video camera, catches video frames which include a visible car number plate and processes them. Once a license plate is detected, its digits are recognized, displayed on the User Interface or checked against a database. The focus is on the design of algorithms used for extracting the license plate from a single image, isolating the characters of the plate and identifying the individual characters.

The Proposed system has been implemented using Vision Assistant 8.5 & LabVIEW 8.5 The performance of the system has been investigated on real images of about 100 vehicles. Recognition of about 85% vehicles shows that the system is quite efficient.

# Chapter 8

## Conclusions and Future Scope

### Conclusions

The process of vehicle number plate recognition requires a very high degree of accuracy when we are working on a very busy road or parking which may not be possible manually as a human being tends to get fatigued due to monotonous nature of the job and they cannot keep track of the vehicles when there are multiple vehicles are passing in a very short time .To overcome this problem, many efforts have been made by the researchers across the globe for last many years. A similar effort has been made in this work to develop an accurate and automatic number plate recognition system.

The objective of this thesis was to study and resolve algorithmic of the automatic number plate recognition systems, such as problematic of machine vision, pattern recognition, and OCR. The problematic has been divided into several chapters, according to a logical sequence of the individual recognition steps. Even though there is a strong succession of algorithms applied during the recognition process, chapters can be studied independently.

This work also contains demonstration RTNPR software, which comparatively demonstrates all described algorithms. I had more choices of programming environment to choose from. We have used Vision assistant 8.5 along with LabVIEW 8.5 to obtain the desired results. RTNPR solution has been tested on static snapshots of vehicles, which has been divided into several catogaries according to difficultness. blurry and skewed snapshots give worse recognition rates than a set of snapshots, which has been captured clearly. The objective of the tests was not to find a one hundred percent recognizable set of snapshots, but to test the invariance of the algorithms on random snapshots systematically.

## **Future Scope**

The implementation works quite well except for the reported software problems. There is still room for improvement through. Here follows some things that can be improved.

A bottleneck of any image processing project has been the camera. The now used camera is sensitive to vibrations and fast changing targets since it has quite shutter times. The optics used do not include auto-focus which would be useful. A good auto-focus makes large focus depth unnecessary, which allows for a larger aperture that shorten the shutter time. A lower focus depth also reducing the background details, which might disturb the algorithm. A good feature reducing the effect of vibrations would be an image stabilizer. An infra-red camera along with an infrared light source would give the best performance and wheather independence would then give a good reflection in the number plate's reflective background.

The implementation has a frame-rate that in some environments is much higher than what is necessary. The complexity of some algorithms could therefore be raised to gain in recognition rate. The selection of ROI could be improved by letting more combinations of segments to be examined. The implementation could may be even let several ROIs be examined parallel until the OCR processing is started. The method of finding the plate in the ROI by searching also be improved.

Also, the issues like stains, smudges, blurred regions & different font style and sizes are need to be taken care of. This work can be further extended to minimize the errors due to them.

# References

- [1] Hu, M. K., "Visual Pattern Recognition by Moment Invariant", IRE Transaction on Information Theory, vol IT- 8, pp. 179-187, 1962.
- [2] Khotanzad, A., and Hong, Y.H., "Invariant image recognition by zeraike moments," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 12, no. 5, pp. 489-497,1990.
- [3] Khotanzad, A., and Hong, Y.H., "Rotation in-variant image recognition using features selected via a systematic method," Pattern Recognition, vol. 23, no. 10, pp. 1089-1101, 1990.
- [4] Belkasim, S.O., Shridhar, M., and Ahmadi, A., "Pattern Recognition with moment invariants: A Comparative study and new results," Pattern Recognition, vol. 24, pp. 1117-1138,1991.
- [5] Lee, E. R., Earn, P. K., and Kim, H. J., "Automatic recognition of a car license plate using color image processing", IEEE International Conference on Image Processing 1994, vol. 2, pp.301-305, 1994.
- [6] Comelli, P., Ferragina, P., Granieri. M. N., and Stabile, F., "Optical recognition of motor vehicle license plates", IEEE Transactions on Vehicular Technology, vol. 44, no. 4, pp: 790-799,1995.
- [7] Morel, J., and Solemini, S., "Variational Methods in Image Segmentation", Birkhauser, Boston, 1995.
- [8] Nieuwoudt, C, and van Heerden, R., "Automatic number plate segmentation and recognition", Seventh annual South African workshop on Pattern Recognition, pp. 88-93, IAPR, 1996.

- [9] Kim, G. M., "The automatic recognition of the plate of vehicle using the correlation coefficient and Hough transform", Journal of Control, Automation and System Engineering, vol. 3, no.5, pp. 511-519, 1997.
- [10] Cho, D. U., and Cho, Y. Ft., "Implementation of pre-processing independent of environment and recognition and template matching ", The Journal of the Korean Institute of Communication Sciences, vol. 23, no. 1, pp. 94-100, 1998.
- [11] Park, S. FL, Kim, K. I., Jung, K., and Kim, H. J., "Locating car license plates using neural network", IEE Electronics Letters, vol.35, no. 17, pp. 1475-1477, 1999.
- [12] Naito, T. Tsukada, T. Yamada, K. Kozuka, K. and Yamamoto, S., "Robust recognition methods for inclined license plates under various illumination conditions outdoors", Proceedings IEEE/IEEJ/JSAI International Conference on Intelligent Transport Systems, pp. 697-702,1999
- [13] Naito, T., Tsukada, T., Yamada, K., Kozuka, K., and Yamamoto, S., "License plate recognition method for inclined plates outdoors", Proceedings International Conference on Information Intelligence and Systems, pp. 304-312, 1999.
- [14] Naito, T. Tsukada, T. Yamada, K. Kozuka, K. and Yamamoto, S., "Robust recognition methods for inclined license plates under various illumination conditions outdoors", Proceedings IEEE/IEEJ/JSAI International Conference on Intelligent Transport Systems, pp. 697-702,1999.
- [15] Salagado, L., Menendez, J. M., Rendon, E., and Garcia, N., "Automatic car plate detection and recognition through intelligent vision engineering", Proceedings of IEEE 33r Annual International Carnahan Conference on Security Technology, pp. 71-76, 1999.
- [16] Naito, T., Tsukada, T., Yamada, K.s Kozuka, K., and Yamamoto, S., "Robust license-plate recognition method for passing vehicles under outside environment", IEEE Transactions on Vehicular Technology, vol: 49 Issue: 6, pp: 2309-2319, 2000.

- [17] Kim, K. K., Kim, K. I., Kim, J.B., and Kim, H. J., "Learning based approach for license plate recognition", Proceedings of IEEE Processing Society Workshop on Neural Networks for Signal Processing, vol. 2, pp: 614-623, 2000.
- [18] Yu, M., and Kim, Y. D., "An approach to Korean license plate recognition based on vertical edge matching", IEEE International Conference on Systems, Man, and Cybernetics, vol. 4, pp. 2975-2980, 2000.
- [19] Yan, Dai., Hongqing, Ma., Jilin, Liu., and Langang, Li, "A high performance license plate recognition system based on the web technique, Proceedings IEEE Intelligent Transport Systems, pp. 325-329, 2001.
- [20] Yan, Dai., Hongqing, Ma., Jilin, Liu., and Langang, Li, "A high performance license plate recognition system based on the web technique, *Proceedings IEEE Intelligent Transport Systems*, pp. 325-329, 2001.
- [21] Hontani, H., and Koga, T., "Character extraction method without prior knowledge on size and information", Proceedings of the IEEE International Vehicle Electronics Conference (IVEC'01), pp. 67-72, 2001.
- [22] Cowell, J., and Hussain, F., "Extracting features from Arabic characters", Proceedings of the IASTED International Conference on COMPUTER GRAPHICS AND IMAGING, Honolulu, Hawaii, USA, pp. 201-206, 2001.
- [23] Hansen, H., Kristensen, A. W., Kohler, M. P., Mikkelsen, A. W. , Pedersen J. M., and Trangeled, M., "Automatic recognition of license plates", Institute for Electronic System, Aalborg University, May 2002.
- [24] Cowell, J., and Hussain, F., "A fast recognition system for isolated Arabic characters", Proceedings Sixth International Conference on Information and Visualisation, IEEE Computer Society, London, England, pp. 650-654, 2002.

[25] Hamami, L., and, Berkani, D., "Recognition System for Printed Multi-Font and Multi-Size Arabic Characters", The Arabian Journal for Science and Engineering, vol. 27, no. IB, pp. 57-72, 2002.

[26] LabVIEW Machine Vision and Image Processing Course Manual

[27] NI Vision Assistant tutorial manual.

[28] NI-IMAQ for USB Cameras User Guide.