VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi – 18

A PROJECT REPORT

on

"AUTOMATIC NUMBER PLATE RECOGNITION SYSTEM"

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Computer Science & Engineering

by

PRANAV S. MAGADI	**1BG12CS070**
SUGOSH N. R.	**1BG12CS100**
SURAJ K.	**1BG12CS105**
VARUN NAGARAJ	**1BG12CS115**

Under the Guidance of

Dr. Sahana D. Gowda
Professor & Head
Dept. of CSE, BNMIT

Vidyaya Amrutham Ashnuthe

B. N. M. Institute of Technology

12th Main, 27th Cross, Banashankari II Stage, Bengaluru - 560 070

Department of Computer Science & Engineering

2015-2016

B. N. M. Institute of Technology

12th Main, 27th Cross, Banashankari II Stage, Bengaluru - 560 070

Department of Computer Science & Engineering

Vidyaya Amrutham Ashnuthe

CERTIFICATE

Certified that the project work entitled "**AUTOMATIC NUMBER PLATE RECOGNITION SYSTEM"** is a bona fide work carried out by

1. PRANAV S. MAGADI	1BG12CS070
2. SUGOSH N.R.	1BG12CS100
3. SURAJ K.	1BG12CS105
4. VARUN NAGARAJ	1BG12CS115

in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science & Engineering** degree of the **Visvesvaraya Technological University**, Belagavi during the year 2015-2016. It is certified that all corrections / suggestions indicated for the internal assessment have been incorporated in the report deposited in the departmental library. The report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Dr. Sahana D. Gowda **Professor & Head** **Dept. of CSE, BNMIT**	**Dr. Sahana D. Gowda** **Professor & Head** **Dept. of CSE, BNMIT**	**Dr. Krishnamurthy G N** **Principal** **BNMIT**

Name of the Examiners	**Signature with date**
1.	
2.	

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned the efforts with success.

We would like to thank the **Management** of **BNM Institute of Technology** for providing such a healthy environment for the successful completion of project work.

We would like to express our deep sense of gratitude to the Director **Professor T. J. Ramamurthy** and the Principal **Dr. Krishnamurthy G N** for their encouragement that motivated us for the successful completion of project work.

It gives us immense pleasure to thank our project guide **Dr. Sahana D. Gowda,** Professor and Head of Department for her constant support and guidance throughout the project work.

We would also like to thank the Project Coordinator **Smt. Savitha G.,** Associate Professor, Department of Computer Science & Engineering and all other teaching and non-teaching staff of Computer Science Department who has directly or indirectly helped me in the completion of the project work.

Last, but not the least, I would hereby acknowledge and thank my parents who have been a source of inspiration and also instrumental in the successful completion of the project work.

Pranav S. Magadi	(1BG12CS070)
Sugosh N.R.	(1BG12CS100)
Suraj K.	(1BG12CS105)
Varun Nagaraj	(1BG12CS115)

ABSTRACT

Automatic Number Plate Recognition System (**ANPRS**) is a mass surveillance method that uses optical character recognition on images to read vehicle registration plates. They can use existing closed-circuit television or road-rule enforcement cameras, or ones specifically designed for the task. They are used by various police forces and as a method of electronic toll collection on pay-per-use roads and cataloguing the movements of traffic or individuals.

The aim of the project work is to design an ANPRs to recognise the number plate of a vehicle captured in a video, which is clustered with other text information on or off the vehicle. Due to climatic and lighting variations, there is a possibility that recognition may be poor. Hence there is a need to identify a technology to enhance the quality of the image, thereby increasing the recognition rate. The work emphasises on implementing the enhancement and recognition techniques.

In this project there are four phases implemented: Pre-processing, Text Localization, Text Segmentation, Text Recognition. In Text localization, various text blocks in image are detected and only the number plate is localized. Text Segmentation extracts individual characters from the localized number plate. Text Recognition phase recognizes the segmented characters and displays it in a new file.

CONTENTS

List of Figures

List of Tables

INTRODUCTION

1.1 Overview

Text mining is a burgeoning new field that attempts to glean meaningful information from natural language text. It may be loosely characterized as the process of analyzing text to extract information that is useful for particular purposes. Compared with the kind of data stored in databases, text is unstructured, amorphous, and difficult to deal with algorithmically. Nevertheless, in modern culture, text is the most common vehicle for the formal exchange of information. The field of text mining usually deals with texts whose function is the communication of factual information or opinions.

The advances in the data capturing, storage, and communication technologies have made vast amounts of video data available to consumer and enterprise applications. Video Analytics, also referred to as Video Content Analysis (VCA), is a generic term used to describe computerized processing and analysis of video streams**.** Computer analysis of video is currently implemented in a variety of fields and industries, however the term “Video Analytics” is typically associated with analysis of video streams captured by surveillance systems.

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

With the rapid development of TV, internet and wireless network, the demand of video indexing and retrieval is increasing. The popularity of digital image and video is increasing rapidly. The texts in video provide an informative clue for video indexing and retrieval because texts carry semantic information more relevant to the video contents than images. Text embedded in multimedia data, as a well-defined model of concepts for human communication, contains much semantic info related to the content. This text information can provide a much truer form of content based access to the image and video documents if it can be extracted and harnessed efficiently.

Generally, video contains two types of text. The first type pertains to caption text. Texts of this type are edited texts or graphics texts artificially superimposed into the video and are relevant to the content of the video. The second type belongs to scene text, which is naturally existing text, usually embedded in objects in the video. Since, caption text is edited according to content of video, it is useful in several applications especially in indexing and retrieving based on semantics. In the same way, scene text is equally important as caption text as it is useful in identifying exciting events in sports video and understanding scenes to study the environment.

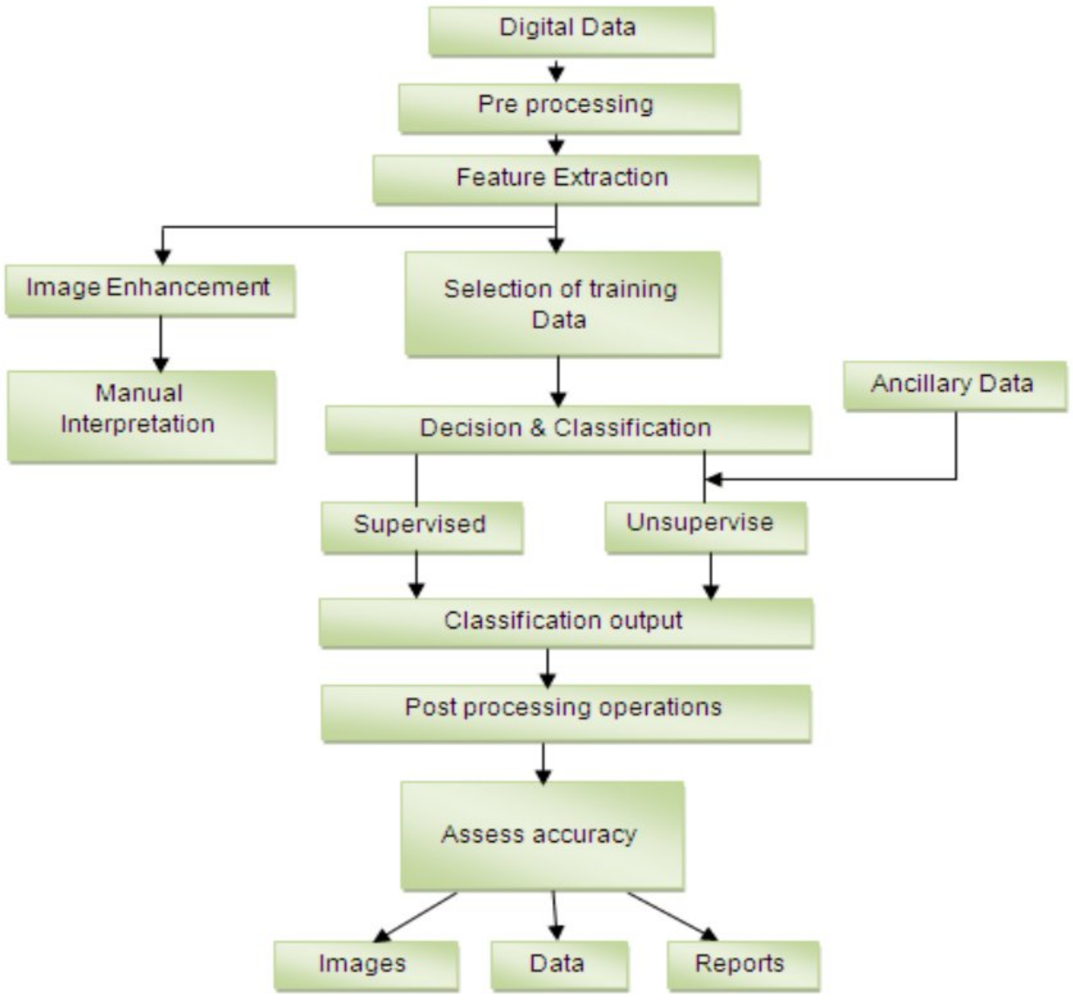


Figure 1.1: Different phases of image processing

In general, the most important and widely used application of text extraction system is text based image indexing and retrieval by using recognized outputs of the system. Besides that many other applications of text extraction system have been developed as well, such as camera based text reading system for visually impaired people, wearable translation robot, vehicle license plate recognition, and mad sign text detection for mobile device.

The three general phases that all types of data have to undergo while using digital technique are pre- processing, enhancement and display, information extraction as shown in the figure 1.1.

A typical image processing scheme, as suggested by various researchers, involves the following four main processes:

[1] **Edge Detection:** Most text has rich edges between its boundaries and the background. Thus some detection of edges is necessary. Hence the transitions between columns for each row are determined. Based on these transitions edges are highlighted.

[2] **Text Localization:** Text localization finds regions in an image that contain text objects. It also groups detected text regions into text objects and generates a set of tight bounding boxes around all text objects.

[3] **Text Segmentation:** The bounding box with the number plate is extracted based on the intensity of white pixels and each character is later segmented.

[4] **Text Recognition:** Text recognition recognizes the characters from the image and displays it in a new file.

1.2 Problem Statement

The aim of the project work is to design an ANPRs to recognize the number plate of a vehicle captured in a video, which is clustered with other text information on or off the vehicle. Due to climatic and lighting variations, there is a possibility that recognition may be poor. Hence there is a need to identify a technology to enhance the quality of the image, thereby increasing the recognition rate. The work emphasizes on implementing the enhancement and recognition techniques.

1.3 Objectives

- To obtain a noise-free binarized image.
- To find contours and put bounding boxes to localise the text.
- To select the most suitable bounding box containing the number plate and extract the text from the detected box.
- Recognise the text and display it in a separate file**.**

1.4 Motivation

The motivation for this project was primarily an interest in undertaking a challenging project in an interesting area of research. The opportunity to learn about a new area of computing was a very enriching experience. The main motivation for doing this project is keen interest in the various fascinating applications of image processing, and how it is entwined in so many areas today. It provides a great research experience and helps us cultivate a deep appreciation for the technology today.

LITERATURE SURVEY

2.1 Background

Image and video, as two most popular multi-media documents, are being produced on a daily basis by a wide variety of sources, including the long distance educational programs, medical diagnostic systems, business and surveillance applications, broadcast and entertainment industries, etc. Recently, with the increasing availability of low cost portable cameras and camcorders, the number of images and videos being captured are growing at an explosive rate. According to the statistics provided by Flickr and YouTube, the quantity of photos uploaded on Flickr has been increasing 40% year-by-year over the last 5 years a and has reached 6 billion in august 2013, and more 13 million hours of videos were uploaded on YouTube during 2013 and 48 hours of hours of video were uploaded every minute. Given such vast quantities of image and video, it is quite probable that most images and videos we are interested in are available and can be accessed somewhere on the internet.

Such huge amount of images and videos make it increasingly difficult for us to locate specific images of interest. Hence there is a need to develop an efficient text extraction algorithm. Towards this goal, much effort has been done and many algorithms have been developed. Fortunately, it can be noted that there is considerable amount of text objects and image and video documents.

2.2 Approaches

A survey on text detection and recognition of images conducted by Qixiang Ye et al [1], analyses, compares, and contrasts technical challenges, methods and performances of text detection and recognition research in color imagery.

Video content analysis (also Video content analytics, VCA) is the capability of automatically analyzing video to detect and determine temporal and spatial events. The algorithm in [2] makes use of resolution enhancement technique which has been employed on low quality video images by using interpolation which generates sharper high resolution image. Texture based text detection and segmentation is applied on the enhanced image. The proposed method is robust enough to detect text regions from low quality video images, and achieves improved precision rate and recall rate.

The algorithm in [3] uses Mexican hat operator for edge detection and Euler number of a binary image for identifying the license plate region. A pre-processing step using median filter and contrast enhancement is employed to improve the character segmentation performance in case of low resolution and blur images. A unique feature vector comprised of region properties, projection data and reflection symmetry coefficient has been proposed.

Mahini et al. [4] extracted license plate candidates using edge statistics and morphological operations and removes the incorrect candidates according to the determined features of license plates.

Chirag N. Paunwala et al. [5]addresses an intrinsic rule-based license plate localization (LPL) algorithm. It first selects candidate regions, and then filters negative regions with statistical constraints. Key contribution is assigning image inferred weights to the rules leading to adaptability in selecting saliency feature, which then overrules other features and the collective measure, decides the estimation.

In [6], a unique feature vector comprised of region properties, projection data and reflection symmetry coefficient has been proposed. Back propagation artificial neural network classifier has been used to train and test the neural network based on the extracted feature.

Texture-based methods [7][8], scan the image at a number of scales, classifying neighborhoods of pixels based on a number of text properties, such as high density of edges, low gradients above and below text, high variance of intensity, distribution of wavelet or discrete convolution theorem (DCT) coefficients, etc.

The limitations of the methods in this category include big computational complexity due to the need of scanning the image at several scales, problems with integration of information from different scales and lack of precision due to the inherent fact that only small (or sufficiently scaled down) text exhibits the properties required by the algorithm. Additionally, these algorithms are typically unable to detect sufficiently slanted text.

Another group of text detection algorithms is based on regions [9][10]. In these methods, pixels exhibiting certain properties, such as approximately constant color, are grouped together. The resulting connected components (CCs) are then filtered geometrically and using texture properties to exclude CCs that certainly cannot be letters. This approach is attractive because it can simultaneously detect texts at any scale and is not limited to horizontal texts.

Blob coloring [11] is another simple and robust technique which some authors such as [12] put it in use to locate characters in the license. Moreover some authors [13] utilized eight-connected version of blob coloring only as a part of license plate localizing procedure.

A wavelet transform based approach in [14][15] have been used. Although these wavelet methods perform well in localization process which finds multiple plates with different orientation angles, but in some cases which the taken image is too far or too near, it might fails.

[16] Proposes to employ the entropy notion to ‘feel’ the text content in a document image without actually reading it, and hence establish the equivalence or otherwise of two corresponding text components. Conventional Entropy Quantifier (CEQ) is being made use of to measure the energy content in the components.

Besides these approaches, several other approaches are proposed in literature which we have not provided herein.

REQUIREMENT ANALYSIS

3.1 Basic Functionality

The application is designed to recognize the text on the number plates of vehicles.

3.2 Requirements

The requirements for a system are the description of the services provided by the system and its operational constraints these requirements reflect the needs of the customer for a system that helps solve the problem.

The application is supposed to have the following attributes/satisfy the following requirements:

1. The application shall take an image as its input.
2. The input image may or may not contain scene text.
3. The input image may contain straight text area(s), where the text can be in small or capital letters or also in the form of numbers.
4. The image may contain text in one or multiple line(s).
5. The image should not be distorted or skewed or contain any overlapping texts.
6. Image should not contain any languages other than what is specified.

3.2.1 Software Requirements

We determine whether a particular software package fits system requirements or not. This section provides the brief description of the software required.

1. Operating System : Windows XP and above, Mac OS, Red Hat Enterprise Linux 6.X
2. Technology : OpenCV, Qt.

3.2.1.1 INTRODUCTION TO OPENCV

Computer vision is the automatic analysis of images and videos by computers in order to gain some understanding of the world. Computer vision is inspired by the capabilities of the human vision system and, when initially addressed in the 1960s and 1970s, it was thought to be a relatively straightforward problem to solve.

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision, originally developed by Intel's research center. The library is cross-platform and free for use under the open-source BSD license.

HISTORICAL OVERVIEW OF OPENCV

Officially launched in 1999, the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, the main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel’s Performance Library Team. In the early days of OpenCV, the goals of the project were described as

- Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
- Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
- Advance vision-based commercial applications by making portable, performance-optimized code available for free—with a license that did not require to be open or free themselves.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000. The second major release of the OpenCV was on October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations. In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site.

OpenCV in Image processing

Image processing is the process of manipulating image data in order to make it suitable for computer vision applications or to make it suitable to present it to humans. For example, changing brightness or contrast is a image processing task which make the image visually pleasing for humans or suitable for further processing for a certain computer vision application.

Computer vision which go beyond image processing, helps to obtain relevant information from images and make decisions based on that information. In other words, computer vision is making the computer see as humans do. Basic steps for a typical computer vision application as follows.

1. Image acquisition
2. Image manipulation
3. Obtaining relevant information
4. Decision making

Applications

- 2D and 3D feature toolkits – 3D reconstruction of images are possible with OpenCV.
- Ego motion estimation - Accurate estimation of the ego-motion of the applications such as a vehicle relative to the road is a key component for autonomous driving and computer vision based products.
- Facial recognition system - A facial recognition system is a computer application capable of identifying or verifying a person from a digital image or a video frame from a video source.
- Gesture recognition - Gesture recognition can be seen as a way for computers to begin to understand human body language, thus building a richer bridge between machines and humans than primitive text user interfaces or even GUIs

- Human–computer interaction (HCI): A user or operator must be able to process whatever information that a system generates and displays; therefore, the information must be displayed according to principles in a manner that will support perception, situation awareness, and understanding.

Programming language

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Perl, and Ruby have been developed to encourage adoption by a wider audience. All of the new developments and algorithms in OpenCV are now developed in the C++ interface.

Hardware Support

OpenCV runs on a variety of platforms. Desktop: Windows, Linux, OSX, FreeBSD, NetBSD, OpenBSD; Mobile: Android, iOS, Maemo, BlackBerry 10. The user can get official releases from SourceForge or take the latest sources from GitHub. OpenCV uses CMake.

Advantages of OpenCV

- **Speed** - OpenCV is basically a library of functions written in C/C++. It's possible to directly provide machine language code to the computer to get executed and therefore, ultimately more image processing done for the computers processing cycles. In OpenCV, we would get at least 30 frames per second, resulting in real-time detection.
- **Resources needed:** Due to the moderate level nature of OpenCV, it uses considerably less systems resources. Typical OpenCV programs only require ~70mb of RAM to run in real-time.
- **Cost:** OpenCV (BSD license) is an open-source product which is free.

3.2.1.2 INTRODUCTION TO QT

Qt is a cross-platform application framework that is widely used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying codebase, while still being a native application with the capabilities and speed thereof. Qt is currently being developed both by the Qt Company, a subsidiary of Digia, and the Qt Project under open-source governance, involving individual developers and firms working to advance Qt. Digia owns the Qt trademark and copyright. Qt is available with both commercial and open source GPL v3, LGPLv3 and LGPL v2 licenses.

PURPOSES AND ABILITIES

Qt is used mainly for developing application software with graphical user interface (GUIs); however, programs without a GUI can be developed, such as command-line tools and consoles for servers. An example of a non-GUI program using Qt is the Cutelyst web framework. GUI programs created with Qt can have a native-looking interface, in which cases Qt is classified as a widget toolkit.

Qt uses standard C++ with extensions including signals and slots that simplifies handling of events, and this helps in development of both GUI and server applications which receive their own set of event information and should process them accordingly. Qt supports many compilers, including the GCC C++ compiler and the visual studio suite. Qt also provides QtSuite, that includes a declarative scripting language called QML that allows using JavaScript to provide the logic. With Qt Quick, rapid application development for mobile devices became possible, although logic can be written with native code as well to achieve the best possible performance. Qt can be used in several other programming language via language bindings. It runs on the major desktop platforms and some of the mobile platforms. It has extensive internationalization support. Non-GUI features include SQL database access, XML parsing, JSON parsing, thread management and network support.

3.2.2 Functional Requirements

For the requirements specification, we use structured natural notation, where the freedom of requirements written is limited and all requirements are written in a standard way.

Automatic Number Plate Recognition system	
Function	Number plate recognition
Description	Localises the text in an image, extracts the characters, recognises them and displays them in a separate file.
Input	Images of vehicles.
Source	Images containing vehicle number plates
Output	Recognised text.
Action	For each image we identify the edges. Based on these edges we draw the bounding boxes and calculate the coordinates of each box. It is then localized and boxes around each character are obtained. We extract the text from the obtained bounding box. The extracted characters are matched with our trained datasets. Later the characters are recognised and printed in a separate file.
Requires	Image dataset of vehicle number plates
Pre-condition	Images should contain vehicle number plates
Post-condition	None

TABLE 3.2.2.1: Software Requirements Specification (SRS)

3.2.3 Non-Functional Requirements

Non-Functional requirements are requirements that specify criteria that can be used to judge the operation of the system, rather than specific behaviour. Functional requirements define what a system is supposed to do whereas the non-functional requirements define how a system is supposed to be.

Some of the non-functional requirements are:

Usability

Simplicity is the key here. The user must be familiar with the user interface and should not have problems migrating to a new system with a new environment. Several users are going to use the system simultaneously, so the usability of the system should not get affected with respect to individual users.

Reliability

The system should be trustworthy and reliable in providing the functionalities. Once a user has made some changes, the changes must be visible by the system. The changes made by the programmer must be visible both to the project manager as well as the test engineer.

Security

Security is a major concern as the organization will have access to various number plates and personal information. An intrusion will result in misuse of personal data.

Scalability

The system should be scalable enough to add new functionalities at a later stage. There should be a common channel, which can accommodate the new functionalities.

Performance

The system is going to be used simultaneously by many employees. Since the system will be hosted on a single web server with a single database server in the background, performance of the system becomes a major concern. The system should not succumb when many users would be using it simultaneously. It should allow fast accessibility to all its users. For example, if two test engineers are simultaneously trying to report the presence of a bug, then there should not be any inconsistency while doing so.

Maintainability

The system monitoring and maintenance should be simple and objective in its approach. There should not be too many jobs running on different machines such that it get difficult to monitor whether the jobs are running without errors.

Portability

The system should be easily portable to another system. This is required when the web server, which is hosting the system, gets stuck due to some problems, which requires the systems to be taken to another system.

Flexibility

The system should be flexible enough to allow modification at any point of time. It should be easy to add new functionalities to the project at any point of time.

Summary

In this chapter we discussed the system requirements and system specification of this project. We discussed various non-functional requirements of the project which is necessary for smooth functioning of the project.

Next chapter gives the overall view of the system design of this project.

SYSTEM ANALYSIS

4.1 Existing System

The text detection and localization approaches are divided into two categories:

(i) Region based approaches that use the different properties between text region and background region to separate text objects. This category typically works in a bottom-up way by separating the image into small regions and then grouping small candidate regions into text regions.

(ii) Texture based approaches that use distinct texture properties of text to extract text from background. This category typically works in a top-down way by extracting texture features of image and then locating text regions.

It is further subdivided into the following approaches based on the features used.

(1) Gradient and Edge Based Approaches:

It is observed that text object typically has high edge densities, similar edge height, big edge gradient magnitudes, and large edge gradient direction variations due to the fact that text is composed of several aligned characters with similar sizes and sharp contrast to background. Gradient and edge-based approaches are implemented using this observation. Usually, morphological operation and block-based dividing and merging are used to generate candidate text regions based on local edge information, and spatial and geometrical constraints are used to remove false alarms. Canny edge detector with two phase thresholding is performed to compute edges in the image.

(2) Colour-Based Approaches:

Color-based approaches are based on the observation that the color of text object is homogenous and different from the background colors. The approaches usually first extract the regions and with homogenous colors based on color similarity using clustering method, intensity histogram, or binarization method, then the candidate text regions are localized by spatial information and geometrical constrains.

(3) Corner-Based Approaches:

Text objects are rich of corners, which are typically uniform distributed over text regions. Based on this property, many corner-based approaches are proposed to separate text from other objects. Block-based corner density and morphological dilation based corner merging are often employed to generate candidate text regions using extracted corners. Harris corners are used to localize text and caption in video document.

(4) Stroke-Based Approaches:

Compared with edge and color features, stroke can be considered as high-level feature because stroke is the prime element of character and contains intrinsic properties of text object. Once stroke information is extracted successfully, text can be localized easily. The most obvious and widely used property of stroke is that the strokes of a character typicaly have similar width and color. However, the approaches based only on stroke width may fail when text objects have more than one dominating stroke widths.

(5) Multiple-Feature Based Approaches:

The multiple-feature based approaches combine the text information captured by different features to detect and localize text objects. It localizes text using edge cue and color cue. Edges are extracted by Sobel detector and filtered by geometric constraints. The edge-based candidate text regions are the CCs obtained using the remaining edges. Color segmentation is done by mean shift segmentation. The color-based candidate text regions are generated using the geometric constraints.

(6) Continuous Line Segment Feature:

The Continuous Line Segment feature extraction method groups together line segments which are continuous and labels them based on their count. Similar line segments are grouped together and identified with a unique label. This will be used by a classifier to train and predict characters.

4.2. Proposed Approach

This project has 4 main tasks for the input image:

1. **Edge Detection:**

   Most text has edges between its boundaries and the background. Thus some detection of edges is necessary. Hence the transitions between columns for each row are determined. Based on these transitions edges are highlighted. Text extraction can be made even more effective by testing whether the presence of an edge is also accompanied by an edge of a color connected component.

2. **Text Localization:**

   Text Localization finds regions in an image that contains text objects. Since we have no a priori information on whether a given image contains text, the first step is to detect whether the text of any sort exists in the frame. This serves as a pre-filtering operation to avoid more expensive processing on all frames and seeks to give a binary answer to the question of whether text exists in the frame. This stage needs to be fast and should prefer false alarm errors to missed detection errors since the former can be rejected at later stages.

3. **Text Segmentation:**

   Text Segmentation stage groups detected text regions into text objects and generates a set of tight bounding boxes around all text objects. Localization ensures that the extraction phase is not only fast but also more efficient since the segmentation is carried out only on small areas of the frame without the distracting information in other areas. Temporal localization is closely related to this stage. It specifies the duration, or lifetime that a particular instance of text remains visible in the frame.

4. **Text Recognition:**

   Text Recognition extracts the text from the image and displays it in a new file. The above text detection procedure only outputs a set of regions in an image, which indicate where in the frame the text appears. Most existing methods for text detection usually end here. To annotate an image using the detected text, it must be extracted. In the following, an efficient method for extracting characters from detected text regions is described.

A text extraction system is composed of the following four stages:

[1] **Edge Detection:** Most text has rich edges between its boundaries and the background. Thus some detection of edges is necessary.

[2] **Text Localization:** Text localization finds regions in an image that contain text objects and localizes the number plate.

[3] **Text Segmentation:** Text Segmentation groups detected number plate into individual text objects and generates a set of tight bounding boxes around all text objects.

[4] **Text Recognition:** Text Recognition recognizes the characters from the image and displays it in a new file.

SYSTEM DESIGN

5.1 Introduction

During analysis, the focus is on what needs to be done, independent of how it is done. During design, decisions are made about how the problem will be solved, first at high level, then at increasingly detailed levels. The designer's goal is to know how the output is to be produced and in what format samples of the output and input are also presented. Second input data and database files have to be designed to meet the requirements of the proposed output. The system designer must make the following decisions:

- Organise the system into subsystems.
- Identify the concurrence inherent in the problem.
- Allocate subsystems to processors and tasks.
- Choose an approach for the management of data stores.
- Handle access to global resources.
- Choose the implementation of the control in software.
- Handle boundary conditions.

The processing phases are handled through the program construction and testing. Finally, details related to justification of system and an estimate of the impact of the candidate system on the user and the organization are documented and evaluated by management as a step toward implementation.

A graphical toll used to describe and analyse the moment of data through a system manual or automated including the process, stores the data, and delays in the system. Data Flow Diagrams are the central tool and basis from which other components are developed. The transformation of data from input to output, through process, may be described logically and independently of the physical components associated with the system. Data flow diagrams are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts.

To design a system in such a way that it separates handwritten text and printed text in a document image and it is split into several steps using image properties.

The system design steps are:

- Pre-processing
- Text localization
- Text segmentation
- Text recognition

Stages of text extraction

Stage 1: Pre-processing

Most images will have some noise present in it. These noises must be removed by using pre-processing operations. Text extraction can be made even more Accept the input image from the dataset. Read the images into the module using the “imread()” function. Resize the image into a standard size of 800 x 800. Apply the Gaussian Blur function to remove sharp edges from the image. The image may contain RGB colors so the image is first converted to black and white by using the “cvtColor” function. Detect all the edges of the frame using the Sobel operator. First, vertical and horizontal edges detected by sobel operator and a “threshold()” function converts the grayscale image into binary. This binary image edges are enhanced by using the Close morphology operation.

Stage 2: Text Localization

Text localization detects the various text blocks in the frame. The different form of text blocks can be slant and/or straight. This can be done by using the “findContours()” function. This function would store the x-y coordinates of all the connected components within the image. Then check if the connected component size is greater than 100. If it is, then convert it into a polygon by using the “approxPolyDP()” function. This polygon is then converted into a standard rectangular bounding box by using “boundingRect()” function.

Stage 3: Text Segmentation

From the bounding boxes obtained from the text localization stage, the boxes with maximum white pixels are indexed and it is used separately by referencing that particular index of the array in which it is stored.

Stage 4: Text Recognition

Text recognition recognises the text from the image and displays it in a new file. The text extraction procedure outputs a set of characters by matching the character codes associated with the set values.

5.2 Data Flow Diagrams

A data flow diagram or DFD is a graphical representation of the “flow” of data through an information system, modelling its process aspects. The DFD is also called as bubble chart. Often they are a preliminary step used to create and overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing or structured designing. DFDs show what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

5.2.1 Description

A Data Flow Diagram (DFD) tracks processes and their data paths within the business or system boundary under investigation. A DFD defines each boundary and illustrates the logical movement and transformation of data within the defined boundary. The diagram shows ‘what’ input data enter the domain, ‘what’ logical processes the domain applies to that data, and ‘what’ output data leaves the domain. Essentially, a DFD is a tool for process modelling and one of the oldest.

5.2.2 Uses of DFDs

A Data Flow Diagram is useful for establishing the boundary or the system domain (the sphere of analysis activity) under investigation.

It identifies any external entities along with their data interfaces that interact with the processes of interest.

A DFD can be a useful tool for helping secure stakeholder agreement (sign-off) on the project scope.

5.3 Levels of Abstraction

In order to fashion an accurate picture of anything, we need to employ various levels of abstraction. This need arises from the basic relationship between an object and an observer, or a subject and its student. There are qualitative changes, as one moves from one level of abstraction to the next, however, and seemingly a progression of form and/ or formative principles. Thus the process of abstraction itself becomes part of the picture, or one of the formative elements thereof.

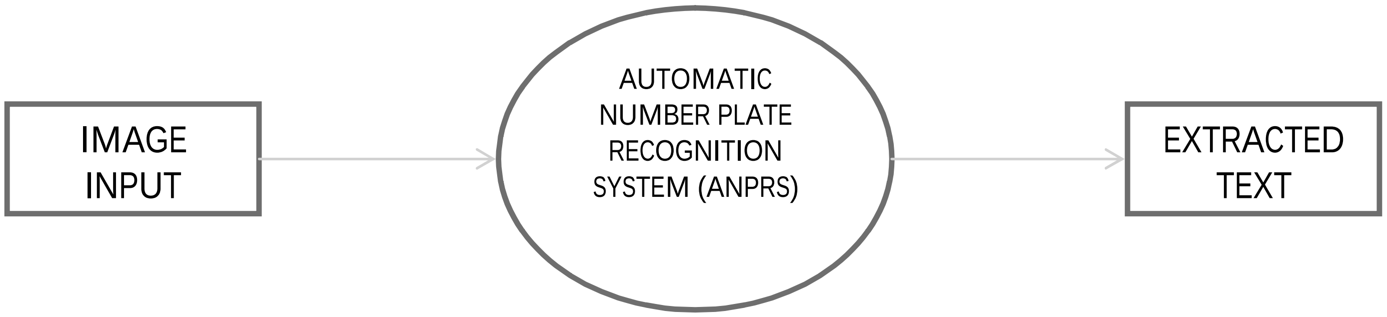


FIGURE 5.3.1: Level 0 DFD

Figure 5.3.1 shows the level 0 DFD. Level 0 DFD will give the overview of the whole system. The input to the system is an image consisting of number plates which will undergo text extraction and the final output of the whole system is the extracted text from the image.

Now when a little closer look to the system is taken then a more detailed DFD is obtained that is the level 1 DFD shown in Figure 5.3.2. First, the system accepts the image as an input. In the text localisation stage the text present in the images are localised and bounding boxes are presented as outputs to the text segmentation stage. In the segmentation stage, each bounding box is broken down into character bounding boxes and is written into a folder. The individual characters are then fed into the text detection module which will detect the characters present and write it as text output.

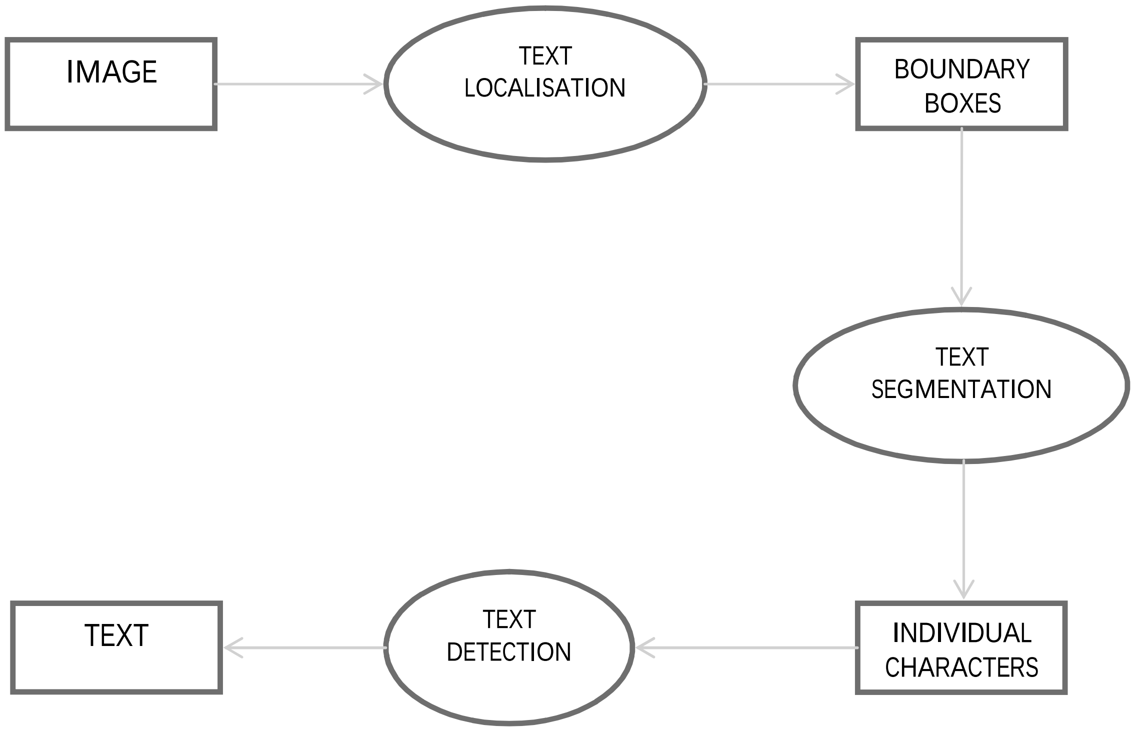


FIGURE 5.3.2: Level 1 DFD

When we go to the next level and take a much closer look into the system, level 2 DFD is obtained, which is shown in 5.3.3. First, the system accepts the input image containing number plates. This image undergoes pre-processing steps of smoothening, noise removal, edge detection and binarization. This module outputs a pre-processed image free of noise and converted into binary. This pre-processed image is then sent to the text localization module where the bounding boxes are drawn around the number plates present in the image. These bounding boxes are then fed into the text segmentation module, where each character present in the number plate is segmented into a separate bounding box.

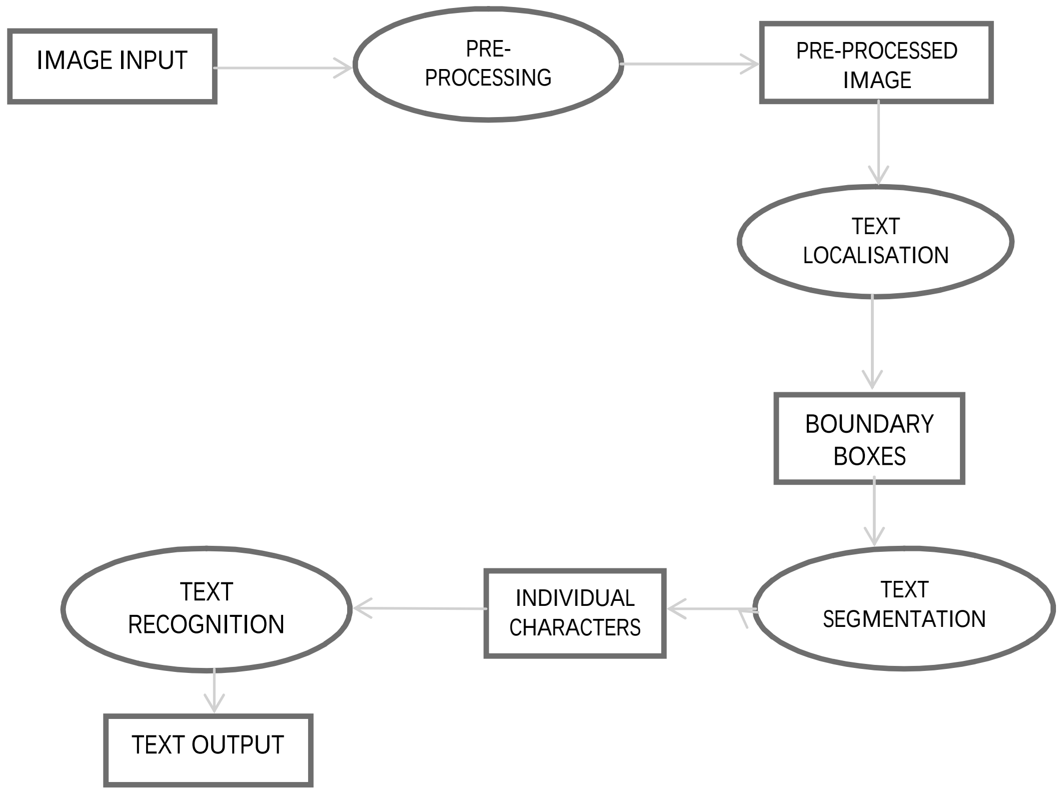


FIGURE 5.3.3: Level 2 DFD

These individual characters are then sent into the text detection module, where the text detection algorithm is implemented and the output of the number plate in the text format is presented.

5.4 Use case diagram

To model a system the most important aspect is to capture the dynamic behavior to clarify a bit in detail. In Unified Modeling Language (UML) there are five diagrams to model dynamic nature. Use case diagram is one of them.

A use case diagram is a representation of user's interaction with the system. A use case diagram can portray the different types if users of a system and the various ways that they interact with the system. In use case diagram, ovals represent processes and arrows represent interaction. Use case diagram should include the following:

- Functionalities to be represented as an use case
- Actors
- Relationships among the use cases and actor

Programmer's View:

1. Programmer loads the image as the input into the system.
2. Then images undergo pre-processing.
3. A set of coordinates of bounding boxes are extracted from each frame and are stored in the database.
4. Using these points, calculate transition values for which we obtain maximum transition.
5. This transitioned image is dilated and merged into single box.

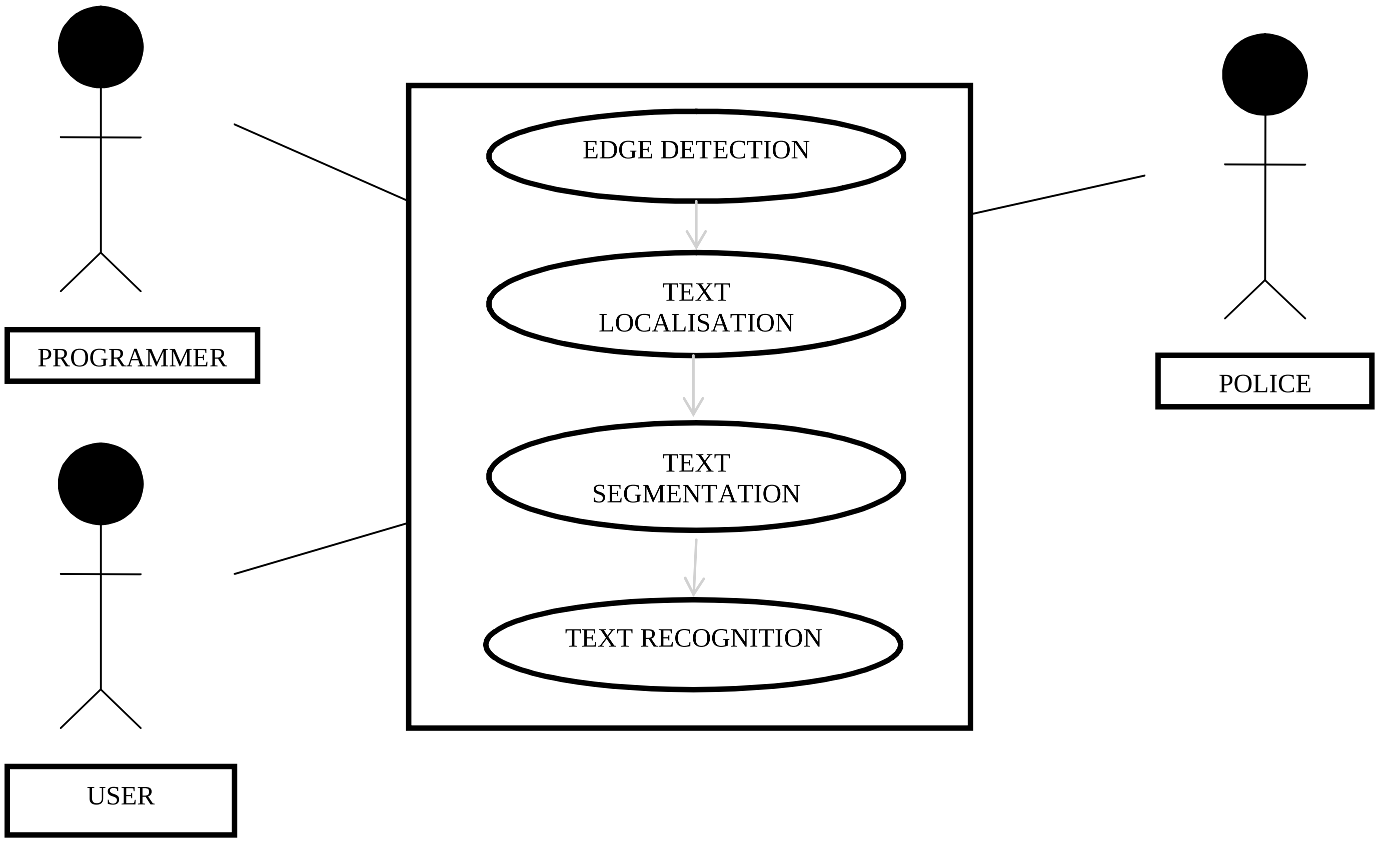


FIGURE 5.4.1: Use Case Diagram

5.5 Sequence Diagram

A sequence diagram also called as event diagram, event scenarios, and timing diagrams shows the object interactions arranged in time sequence. It depicts the objects and classes resolved in the scenario and the sequence of messages exchanged between the objects processed to carry out the functionality of the scenario.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously and as horizontal arrows, the messages exchanged between them, in the order in which they occur.

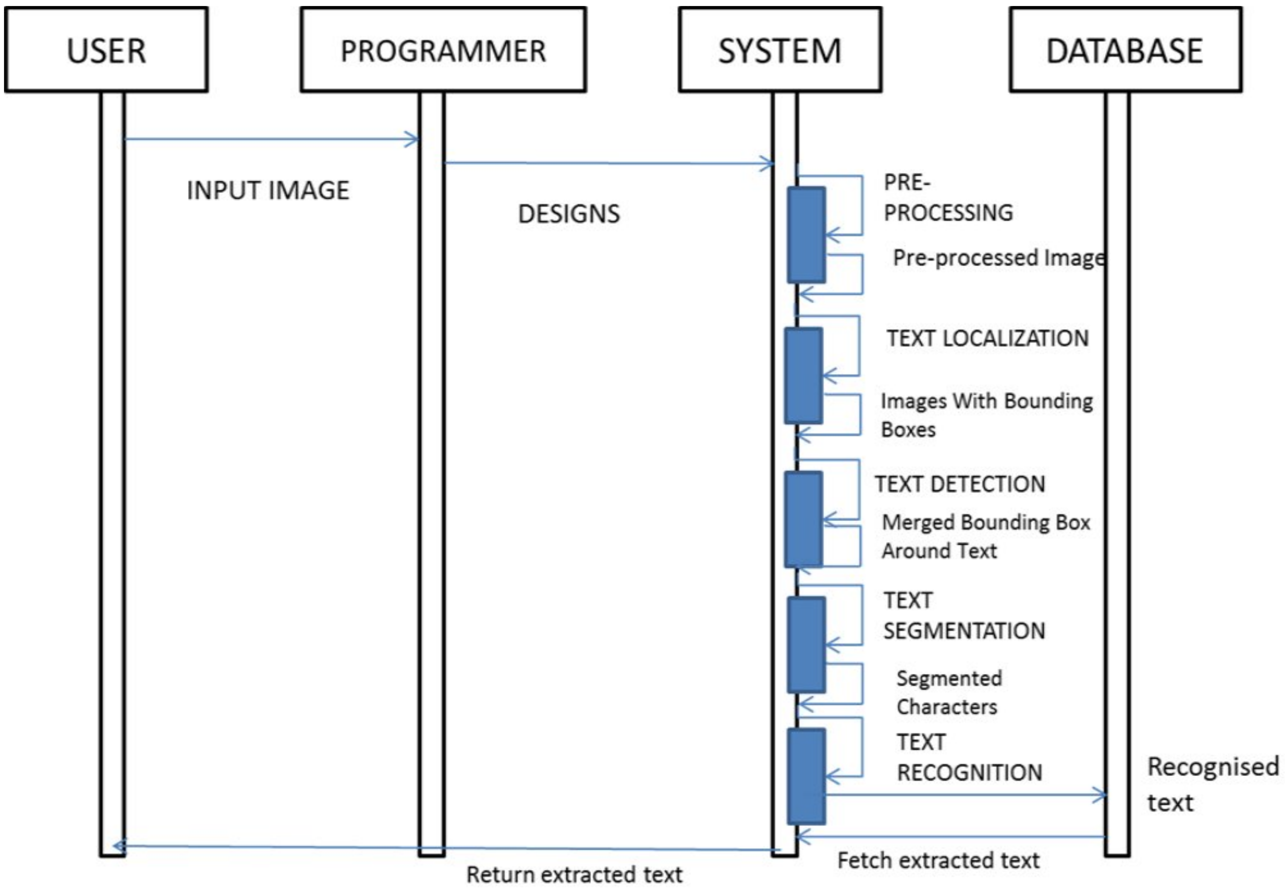


FIGURE 5.5.1: Sequence Diagram

Steps followed in the sequence diagram:

1. Users input the image to the programmer.
2. The programmer loads the dataset on to the system.
3. The system internally performs pre-processing on the image and provides a edge detected output to Text localization.
4. The system performs text localization on the edge detected image to obtain localized number plate by applying bounding box around it.
5. The system performs Text segmentation on the localized number plate to obtain individual characters which will be recognized later.
6. The system performs Text recognition by classifying the segmented characters based on trained data from the database and output a file with recognized text.
7. The user then uses the recognized text to analyse.

IMPLEMENTATION

We begin the implementation with edge detection, which highlights the edges and draws a bounding box around it using text localisation module. A bounding box is drawn across the connected components and the box containing the most number of white pixels is selected. The box is most likely to contain the number plate. The box is later subjected to segmentation where each character in the plate is extracted. The extracted characters are later recognised through various approaches and are printed in a separate file.

6.1 Algorithm and detailed description for each module

The five major modules used in this project are:

- Pre-Processing
- Text localisation
- Text segmentation
- Feature extraction
- Classification

The final two modules constitute the text recognition phase.

6.1.1 Pre- processing

Accept the image and convert that into grayscale image. The canny edge detection module is called and the text regions along with region with sharp edges are detected.

ALGORITHM

Input: RGB Image

Output: Edge detected area

Step 1: Start

Step 2: Apply Gaussian operator

Step 3: Convert image to grayscale

Step 4: Call Sobel edge detection module

Step 5: Convert image to binary format

Step 6: Stop

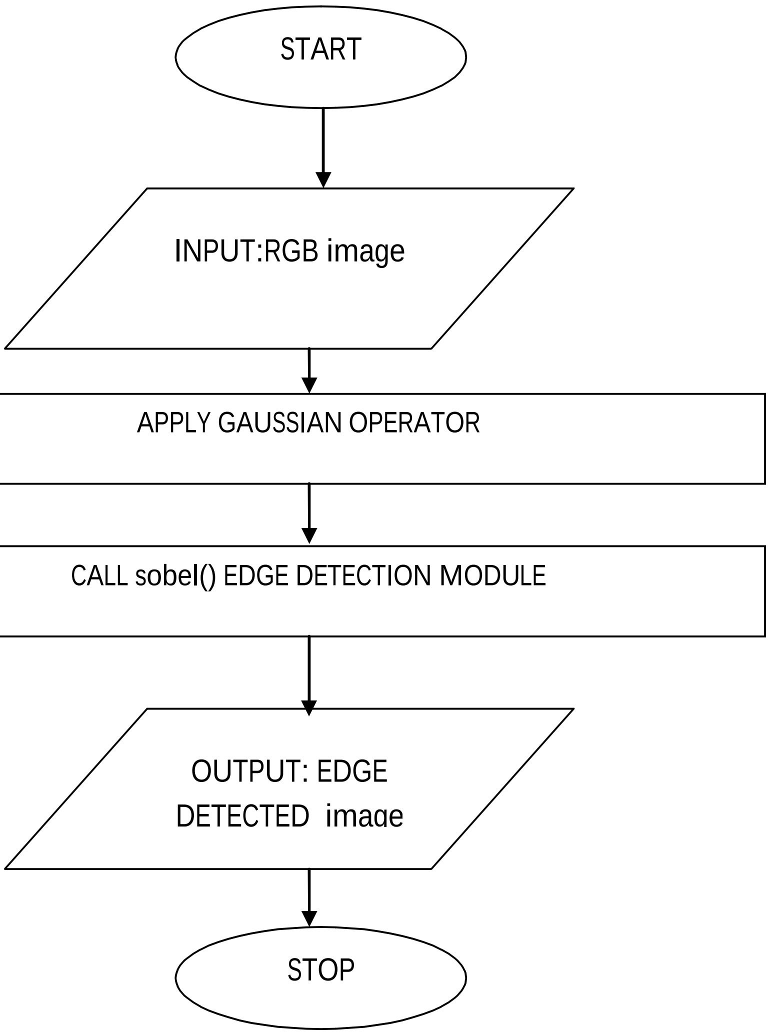

START
INPUT:RGB image
APPLY GAUSSIAN OPERATOR
CALL sobel() EDGE DETECTION MODULE
OUTPUT: EDGE DETECTED image
STOP

FIGURE 6.1.1: Flowchart of Pre-Processing

6.1.2 Text Localisation

The morphology function Close() is applied on the binarized image. findcontours() is later called which retrieves contours from the binary image. The contours are a useful tool for shape analysis and object detection and recognition. A bounding box is applied to each contour.

<u>ALGORITHM</u>

Input: Binarized Image

Output: Image with bounding boxes

Step 1: Start

Step 2: Apply Close Morphology function.

Step 3: Use findcontours() function to find the connected components.

Step 4: Apply the bounding boxes around each contour.

Step 5: Stop

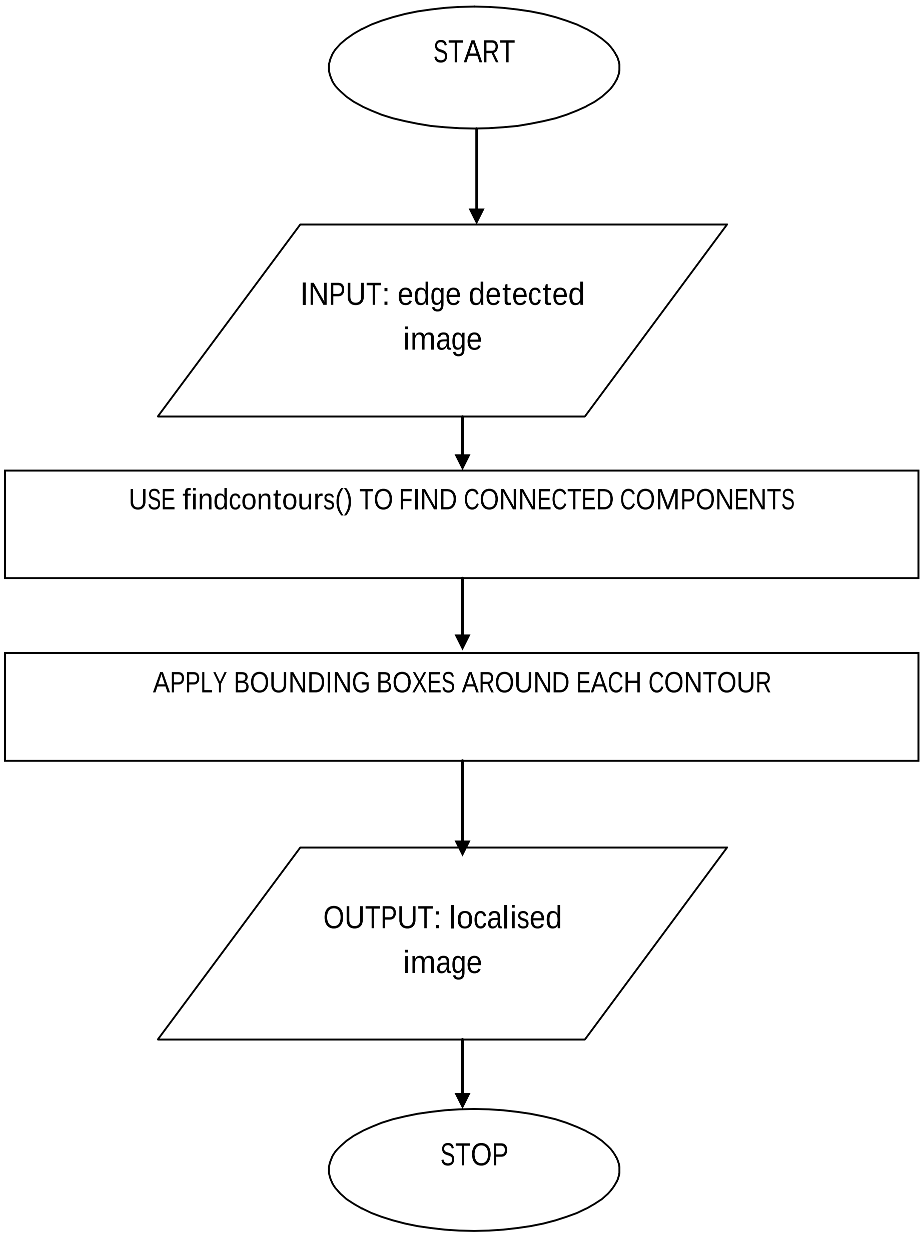

START
INPUT: edge detected image
USE findcontours() TO FIND CONNECTED COMPONENTS
APPLY BOUNDING BOXES AROUND EACH CONTOUR
OUTPUT: localised image
STOP

FIGURE 6.1.2: Flowchart of Text Localisation

6.1.3 Text Segmentation

Segmentation finds the bounding box which contains the number plate. Since there would be many bounding boxes in the localised frame, the boxes that do not contain text need to be eliminated. Constraints are applied to determine the exact coordinates of the bounding boxes. The bounding box

ALGORITHM

Input: bounding boxes.

Output: segmented characters.

Step 1: Select the bounding box with most number of white pixels.

Step 2: Apply findContour() function to the selected bounding box.

Step 3: Apply approxPolyDP() function to make the bounding box into an approximate polygonal shape.

Step 4: Next apply boundingRect() function to convert the polygonal boxes into standard rectangular shapes.

Step 5: Each character is thus extracted into a separate window.

6.1.4 Feature extraction 1

In feature extraction-1, horizontal, vertical and sideways projections are applied to each character. Slopes for each point is calculated and the points are neglected if there is no variation in the slopes i.e., if there is no variation in slope from +ve to -ve then the points are retained. Codes based on angle of deviation for each point is assigned and these codes are fed as inputs to the next phase.

ALGORITHM

Input: Segmented characters.

Output: Unique Code for each character.

Step 1: Apply horizontal, vertical and sideways projection on each extracted character.

Step 2: Find the slope for each point on the character. If there is a variation in slope from +ve to –ve or vice versa then retain those points.

Step 3: Assign codes based on angle of deviation for each point. Study these points for 40 datasets of each character.

Step 4: Use the obtained codes as input to the classification algorithm.

6.1.5 Feature extraction 2

The energy distribution along the transition in the image is noted. This is known as Entropy. Conventional energy quantifier measures the energy distribution of each row and column by considering the probable occurrence of positive and negative transitions among the total number of pixels in each row

<u>ALOGRITHM</u>

Input: Extracted Character.

Output: Entropy Values.

Step 1: For each character extracted, find the column wise and row wise +ve and –ve transitions.

Step 2: Calculate the entropy for each character by using the following formula.

$E(t)= p \log(1/p) + (1-p)\log(1/1-p)$

Step 3: Find the total entropy for each row and column as the summation of the individual entropies of $E^{+}(t)$ and $E^{-}(t)$

6.1.6 Classification

In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.

<u>ALGORITHM</u>

Input: Codes obtained for each character

Output: Recognised characters.

Step 1: Label the gradient codes obtained to their respective characters

Step 2: Train a non-linear (RBF) Support Vector Machine by feeding the gradient codes which are labelled in the previous step to predict the future occurrences of same character.

6.2 Commands Used

imread( )

Syntax :Mat imread(const String& **filename**, int **flags**=IMREAD_COLOR)
Parameters :
filename – Name of file to be loaded.

flags –Flags specifying the color type of a loaded image:

CV_LOAD_IMAGE_ANYDEPTH - If set, return 16-bit/32-bit image when the input has the corresponding depth, otherwise convert it to 8-bit.

CV_LOAD_IMAGE_COLOR - If set, always convert image to the color one
CV_LOAD_IMAGE_GRAYSCALE - If set, always convert image to the grayscale one
Description :The function imread loads an image from the specified file and returns it. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format), the function returns an empty matrix.

imshow( )

Syntax :void imshow(const string& **winname**, InputArray **mat**)

Parameters :

winname – Name of the window

image – Image to be shown

Description :The function imshow displays an image in the specified window. If the window was created with the CV_WINDOW_AUTOSIZE flag, the image is shown with its original size, however it is still limited by the screen resolution. Otherwise, the image is scaled to fit the window.

resize( )

Syntax :void resize(InputArray **src**, OutputArray **dst**, Size **dsize**, double **fx**=0, double **fy**=0)

Parameters :

src – input image.

dst – output image; it has the size dsize (when it is non-zero) or the size computed from src.size(), fx, and fy; the type of dst is the same as of src.

dsize –output image size; if it equals zero, it is computed as:

Either dsize or both fx and fy must be non-zero.

fx –scale factor along the horizontal axis; when it equals 0.

fy –scale factor along the vertical axis; when it equals 0.

Description : The function resize resizes the image src down to or up to the specified size. The initial dst type or size are not taken into account.

GaussianBlur( )

Syntax :void GaussianBlur(InputArray **src**, OutputArray **dst**, Size **ksize**, double **sigmaX**, double **sigmaY**=0, int **borderType**=BORDER_DEFAULT)

Parameters :

src – input image; the image can have any number of channels, which are processed independently.

dst – output image of the same size and type as src.

ksize – Gaussian kernel size. ksize.width and ksize.height can differ but they both must be positive and odd.

sigmaX – Gaussian kernel standard deviation in X direction.

sigmaY – Gaussian kernel standard deviation in Y direction

If sigmaY is zero, it is set to be equal to sigmaX, if both sigmas are zeros, they are computed from ksize.width and ksize.height , respectively; to fully control the result regardless of possible future modifications of all this semantics, it is recommended to specify all of ksize, sigmaX, and sigmaY.

borderType – pixel extrapolation method.

Description : This function blurs an image using a Gaussian filter. The function convolves the source image with the specified Gaussian kernel.

Sobel ( )

Syntax: void Sobel(InputArray **src**, OutputArray **dst**, int **ddepth**, int **dx**, int **dy**, int **ksize**=3, double **scale**=1, double **delta**=0, int**borderType**=BORDER_DEFAULT)

Parameters:

src – input image.
dst – output image of the same size and the same number of channels as src .
ddepth –

output image depth; the following combinations of src.depth() and ddepth are supported:

- src.depth() = CV_8U, ddepth = -1/CV_16S/CV_32F/CV_64F
- src.depth() = CV_16U/CV_16S, ddepth = -1/CV_32F/CV_64F
- src.depth() = CV_32F, ddepth = -1/CV_32F/CV_64F
- src.depth() = CV_64F, ddepth = -1/CV_64F

when ddepth=-1, the destination image will have the same depth as the source; in the case of 8-bit input images it will result in truncated derivatives.

xorder – order of the derivative x.
yorder – order of the derivative y.
ksize – size of the extended Sobel kernel; it must be 1, 3, 5, or 7.
scale – optional scale factor for the computed derivative values; by default, no scaling is applied (see getDerivKernels()for details).
delta – optional delta value that is added to the results prior to storing them in dst.
borderType – pixel extrapolation method (see borderInterpolate() for details).

Description: The function finds edges in the input image image and marks them in the output map edges using the Sobel algorithm. The Sobel operators combine Gaussian smoothing and differentiation, so the result is more or less resistant to the noise. Most often, the function is called with (xorder = 1, yorder = 0, ksize = 3) or (xorder = 0, yorder = 1, ksize = 3) to calculate the first x- or y- image derivative.

cvtColor( )

Syntax :void cvtColor(InputArray **src**, OutputArray **dst**, int **code**, int **dstCn**=0)

Parameters :

src – input image: 8-bit unsigned, 16-bit unsigned, or single-precision floating-point.
dst – output image of the same size and depth as src.
code – color space conversion code
dstCn – number of channels in the destination image; if the parameter is 0, the number of the channels is derived automatically from src and code .

Description :The function converts an input image from one color space to another. In case of a transformation to-from RGB color space, the order of the channels should be specified explicitly (RGB or BGR). Note that the default color format in OpenCV is often referred to as RGB but it is actually BGR (the bytes are reversed). So the first byte in a standard (24-bit) color image will be an 8-bit Blue component, the second byte will be Green, and the third byte will be Red. The fourth, fifth, and sixth bytes would then be the second pixel (Blue, then Green, then Red), and so on.

threshold( )

Syntax :double threshold (InputArray **src**, OutputArray **dst**, double **thresh**, double **maxval**, int **type**)

Parameters :

src – input array (single-channel, 8-bit or 32-bit floating point).
dst – output array of the same size and type as src.
thresh – threshold value.
maxval – maximum value to use with the THRESH_BINARY thresholding types which can be seen in Equation 6.1
type – thresholding type

There are several types of thresholding supported by the function. They are determined by type as shown below. The maximum threshold value is set by the variable termed as maxval.

THRESH_BINARY

$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{maxval} & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

Description :The function applies fixed-level thresholding to a single-channel array. The function is typically used to get a bi-level (binary) image out of a grayscale image (compare()) could be also used for this purpose) or for removing a noise, that is, filtering out pixels with too small or too large values.

adaptiveThreshold( )

Syntax : void adaptiveThreshold(InputArraysrc, OutputArraydst, double maxValue, intadaptiveMethod, intthresholdType, intblockSize, double C)

Parameters :

src – Source 8-bit single-channel image.

dst – Destination image of the same size and the same type as src .
maxValue – Non-zero value assigned to the pixels for which the condition is satisfied. See the details below.
adaptiveMethod – Adaptive thresholding algorithm to use:
ADAPTIVE_THRESH_MEAN_C
or ADAPTIVE_THRESH_GAUSSIAN_C
thresholdType – thresholding type that can be THRESH_BINARY.
blockSize – Size of a pixel neighborhood that is used to calculate a thresholding value for the pixel: 3, 5, 7, and so on.
C – Constant subtracted from the mean or weighted mean normally, it is positive but may be zero or negative as well.
Description : The function transforms a grayscale image to a binary image.

RESULTS

7.1 PRE-PROCESSING

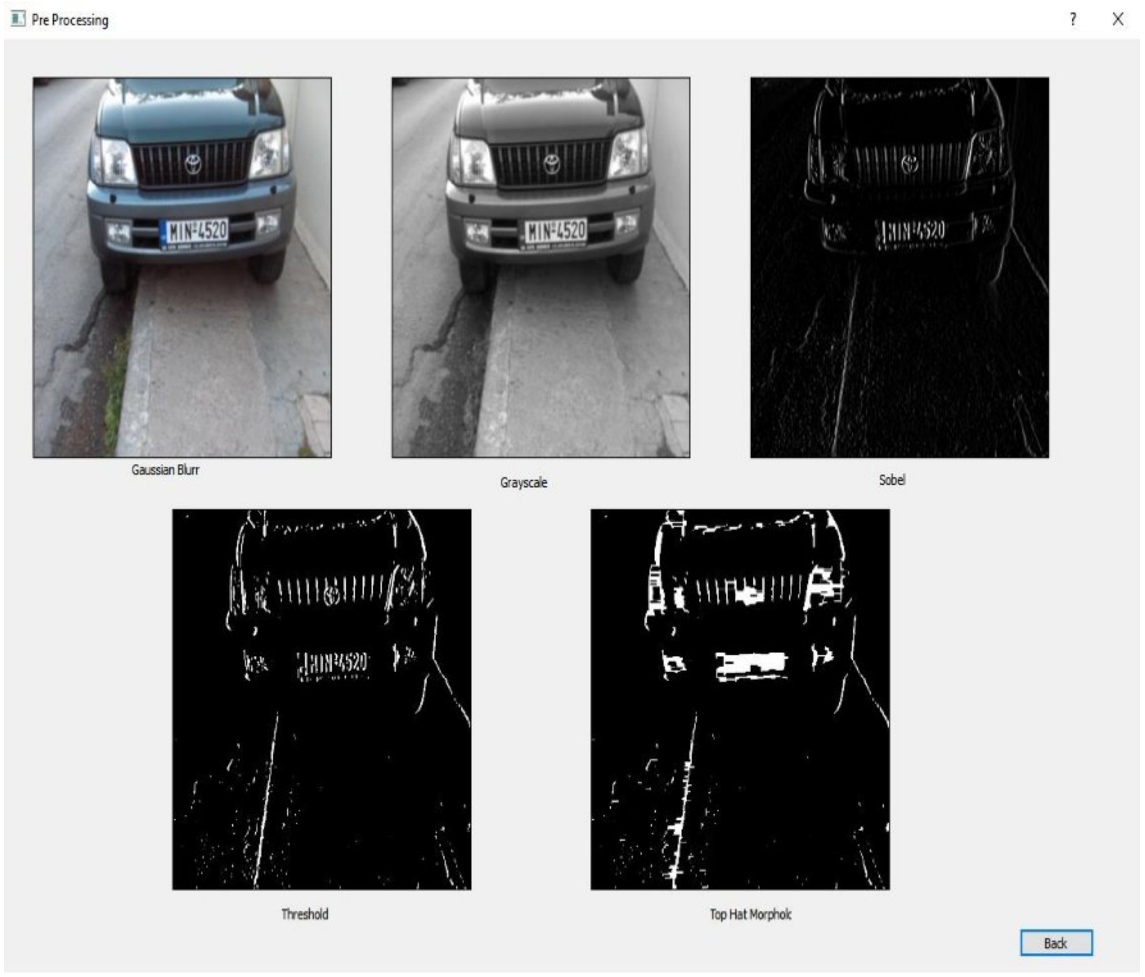


Figure 7.1.1: Pre-processing-1

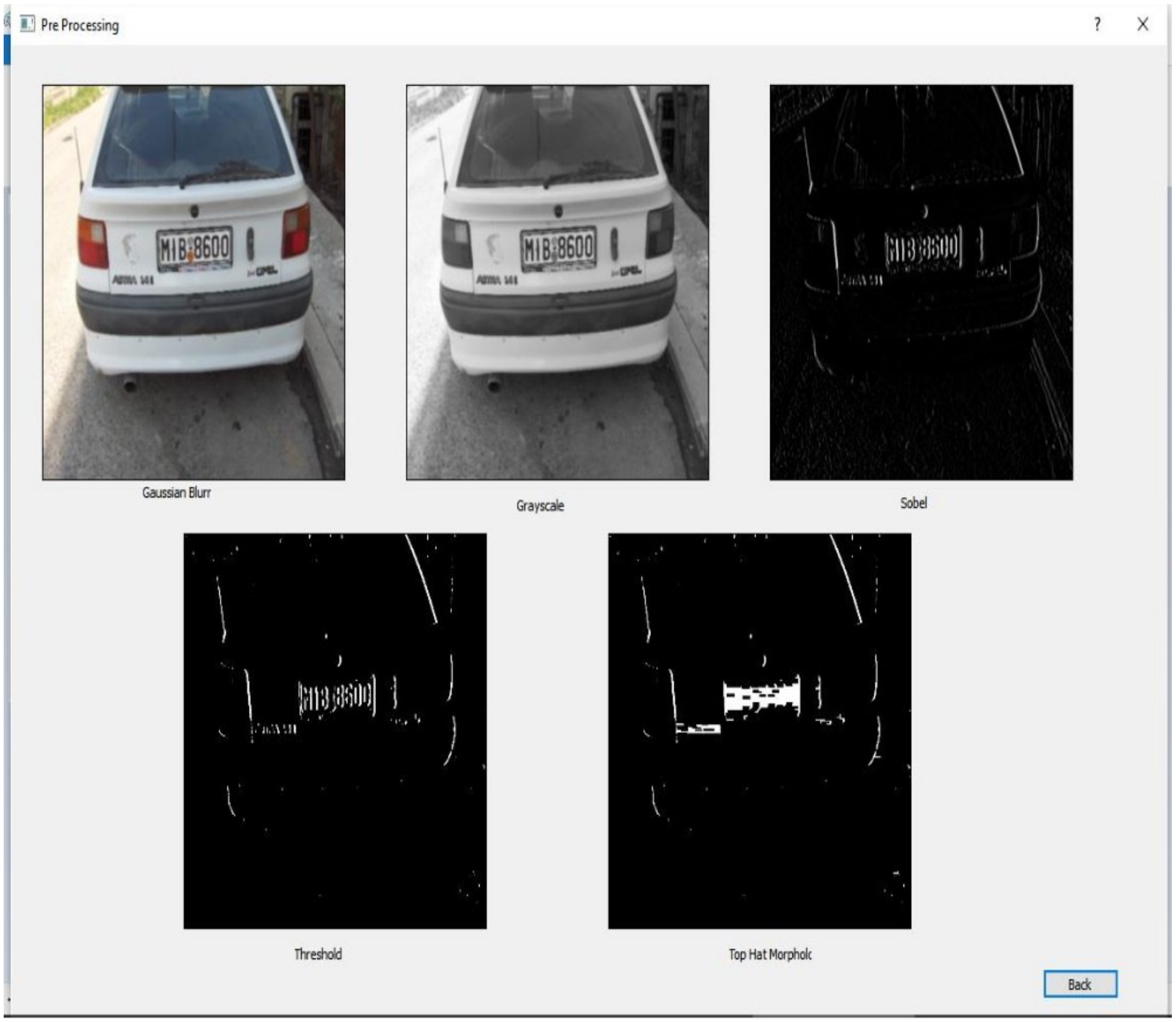


Figure 7.1.2: Pre-processing-2

The Figure 7.1.2 shows the various outputs of the pre-processing stages. The first output image shows the result after a Gaussian Blur operator is used on the input image to smoothen the edges. On applying the Grayscale function the RGB image is converted into a grayscale image as shown in the second output. After converting the image to grayscale, Sobel edge detection operator is applied which results in the third output. This edge detected output is followed by an application of threshold to filter out unnecessary edges, leaving behind only the number plate. This output is presented as input to the Tophat morphological operator, which finds the difference between the input image and its opening.

7.2 TEXT LOCALIZATION

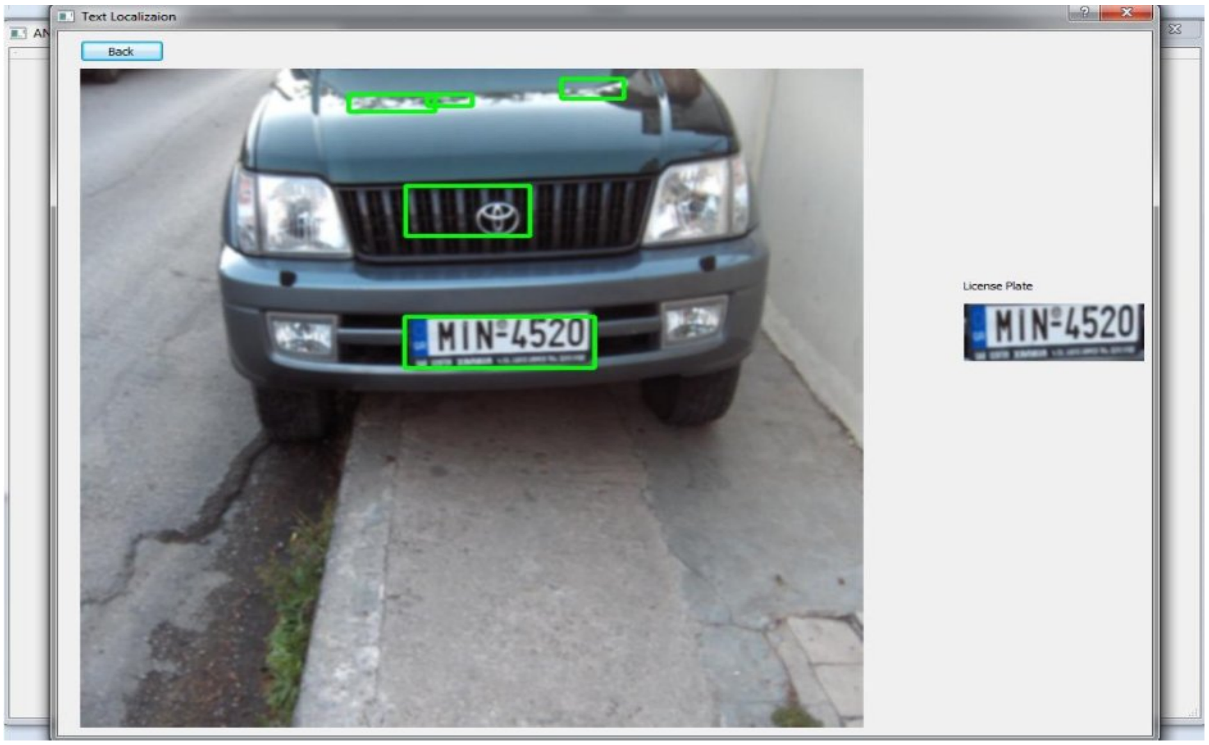


Figure 7.2.1: Text Localization-1

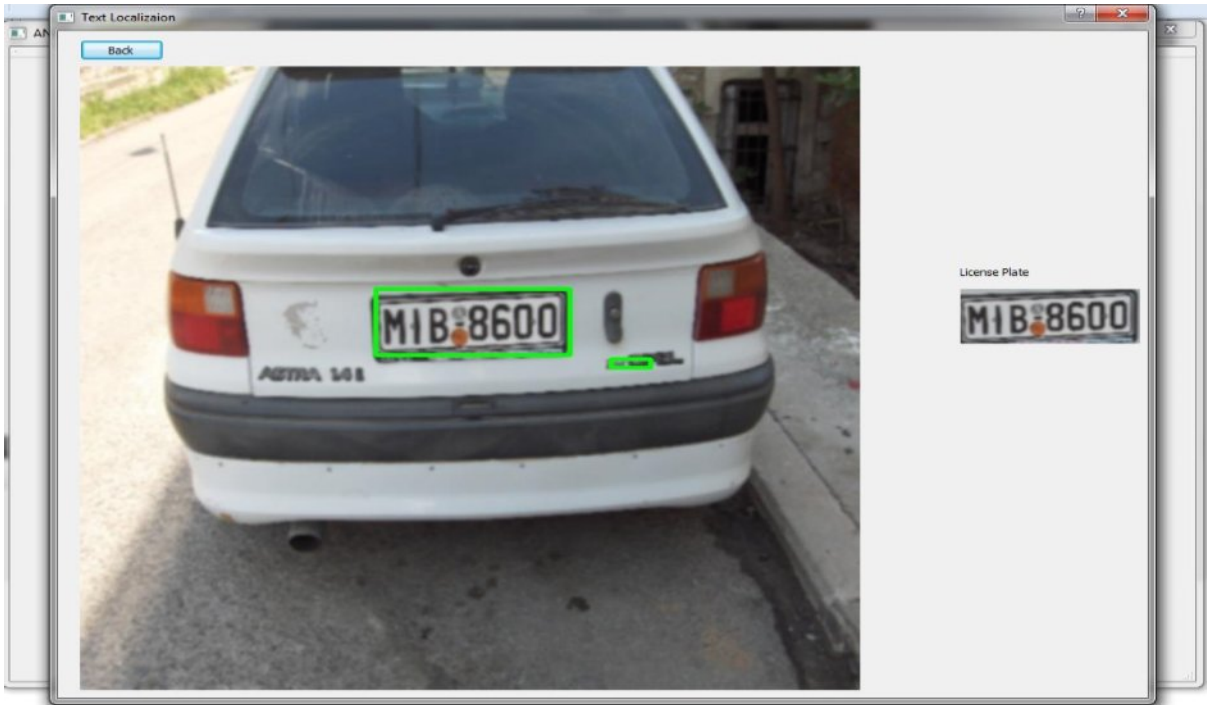


Figure 7.2.2: Text localization-2

7.3 TEXT SEGMENTATION

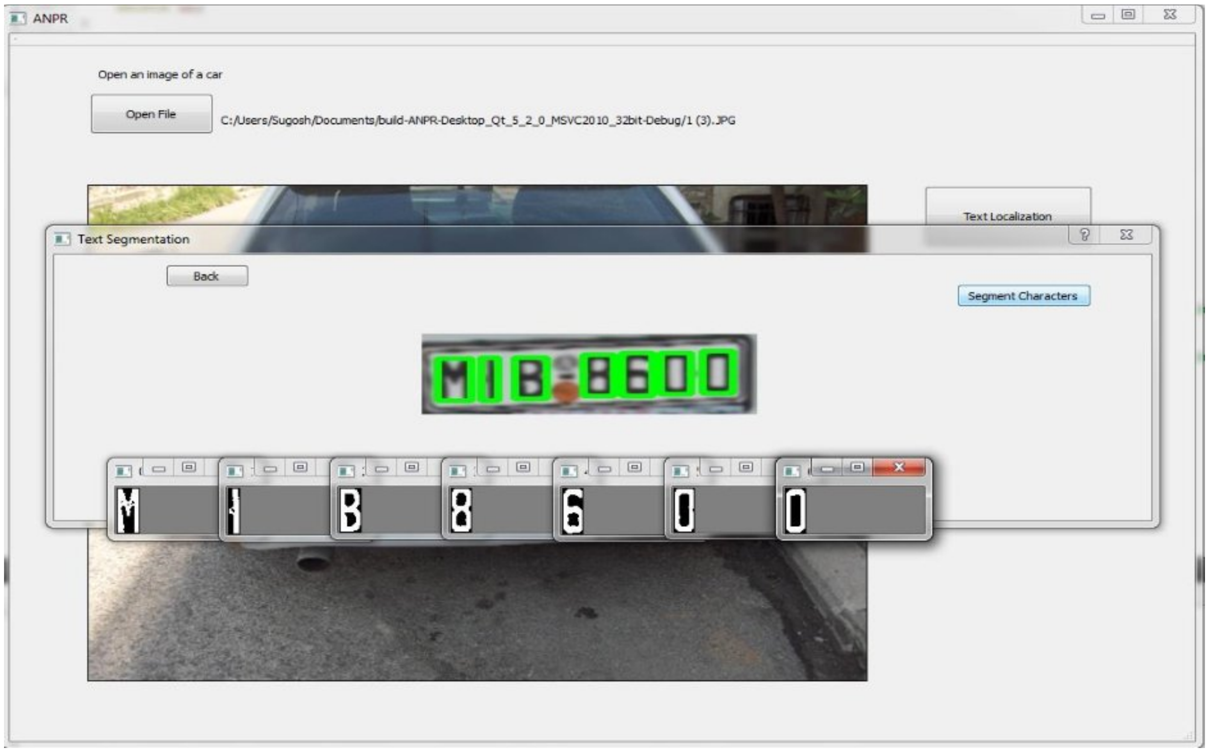
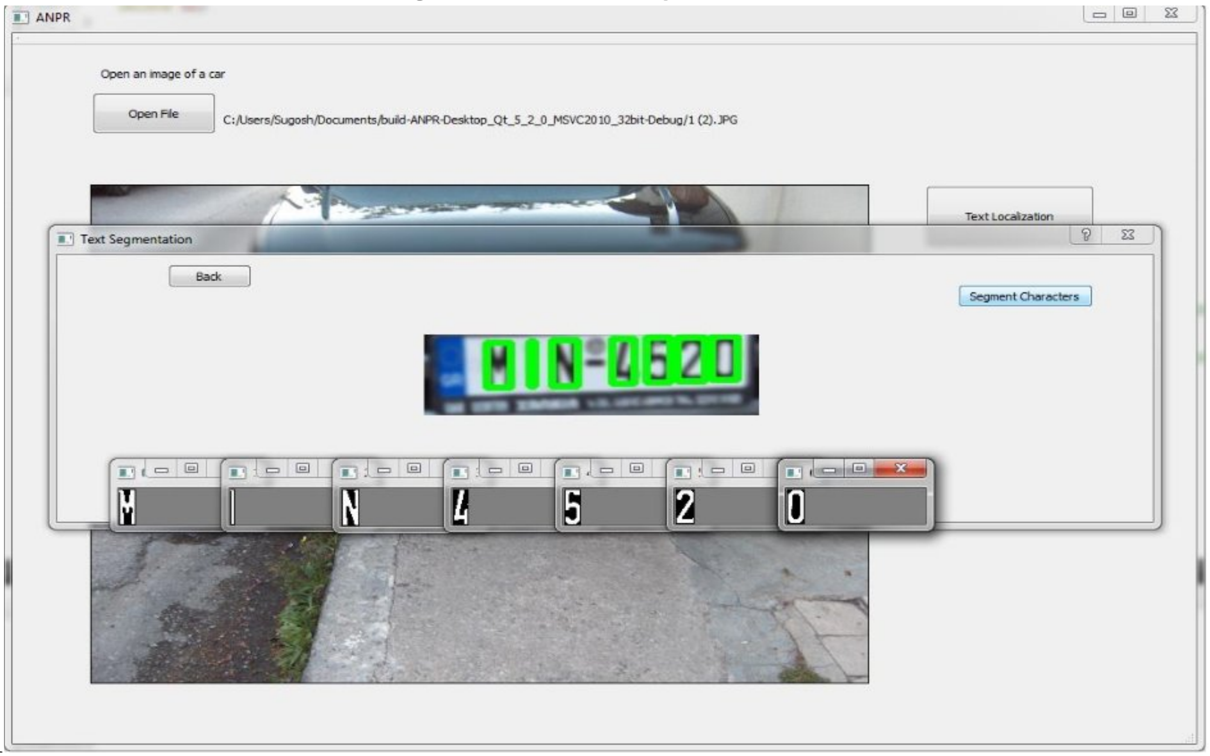


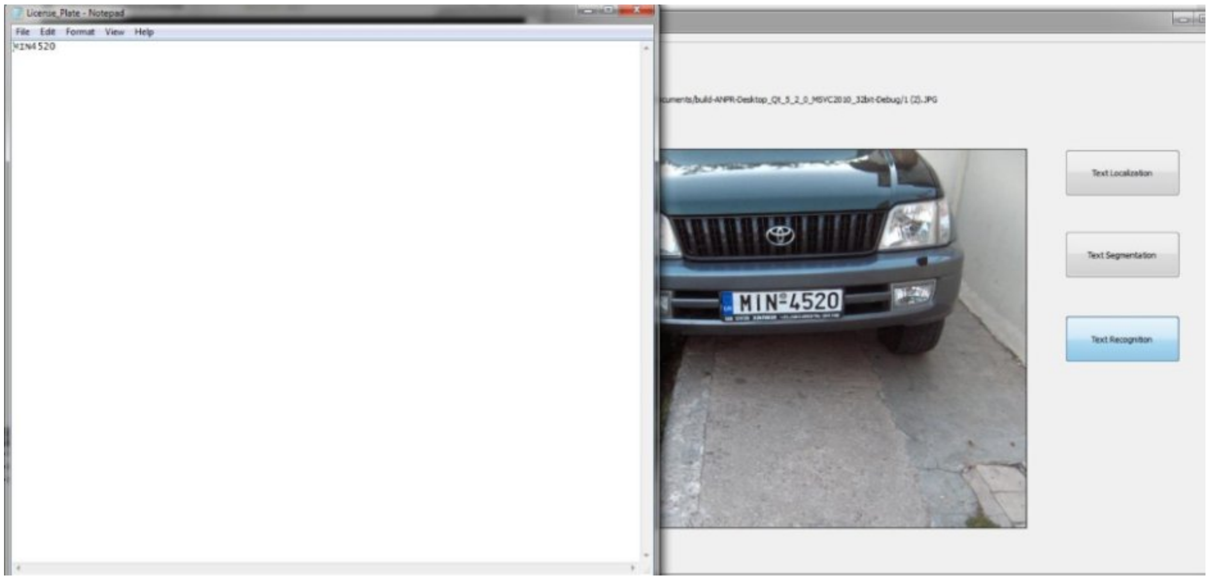
Figure 7.3.1: Text segmentation-



1

Figure 7.3.2: Text Segmentation-2

7.4 TEXT RECOGNITION



```
MIN4520
```

```
cuments/build-ANPR-Desktop_Qt_5_2_0_MSVC2010_32bit-Debug/1 (2).JPG
```

Figure 7.4.1: Text Recognition

TESTING

8.1 Introduction

The most important phase in any developing software is testing. Testing is the final verification and validation activity within the organization itself. Software testing is the technical investigation of the product under test, to provide stakeholders with quality related information. In particular, software system testing is a method of verification involving systematically executing software, to detect defects.

For testing the objective should be to design a test that systematically uncovers different classes of errors and to do so with minimum amount of time and effort. Testing is carried out during the implementation phase to verify that the software behaves as intended by its designer and after the implementation phase is complete. Later testing phase confirms with the requirements of the system.

The project deals with extraction of text blocks and another crucial aspect of this project. It will store co-ordinates of each bounding box in the main input image and draw the bounding box of each word.

8.2 Testing Objectives

The main objective of testing is to prove that the software product meets a set of pre-established acceptance criteria under a prescribed set of environmental circumstances. Another objective is to prove that the design and coding correctly respond to the requirements.

8.3 Testing Strategies

The following strategies are used to depict the error or result of modules of the project.

8.3.1 Unit Testing

Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, individual function or procedure. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process.

In this implemented project,

- The module edge detection was tested for different images and input parameters were suitably adjusted to get segmented frames and edges were detected for each image.
- The module text localization was tested with segmented images to get the desired bounding boxes.
- The module text detection was tested by merging the bounding boxes and obtaining box around the text.
- The module text extraction was done and tested to get only the text object in a separate file

8.3.2 Integration Testing

The ultimate aim of integration testing is to verify whether all modules when integrated are working properly. The integration of all the modules satisfies the aim of the project. The modules which are interactive with one another were clustered and tested together. Take unit tested methods and build a program structure that has been dictated by the design.

The program to extract text from video consisting of 4 modules(edge detection, text localization, text detection and text extraction) were tested together for different videos consisting of text with complex background and the results were noted down. The text objects were extracted from the image and displayed.

8.4 Test Cases

Test Case ID	Test Case Description	Input	Expected Output	Actual Output	Remarks
1	Conversion to Binary Image	Color Image	A black and white image of the color image that has been given as input	A black and white image	PASS
2	Edge Detection	Binary Image	Edges are detected in the frame	Edges are detected in the frame	PASS
3	Text Localization	Detected Edges in the Image	Bounding box around these edges	Bounding box around these edges	PASS
4	Text Detection	Frame containing bounding boxes	Bounding box around the text	Bounding box around the text	PASS
5	Text Segmentation	Image containing bounding boxes around the text	Individual characters segmented from detected text	Segmented characters obtained	PASS
6	Text Recognition	Codes obtained for each character	Recognized letters	80% accuracy. Fails when skewed and blurred images are input	PASS

TABLE 8.4.1: Test Cases

CONCLUSION

Text in an image is the easiest way of understanding the scenario. However, extracting text from number plates is a very difficult task due to the varying font, size, color, orientation, and deformation of text objects.

In the implemented project we have considered text in the number plate as input. The images containing the number plate is subjected to edge detection, text localization, text segmentation and finally recognition. The recognized text is later displayed in a separate text file.

The system works satisfactorily for wide variations in illumination conditions and different types of number plates commonly found in India. It is definitely a better alternative to the existing manual systems in India

LIMITATIONS AND FUTURE ENHANCEMENTS

LIMITATIONS

The limitations that we come across in this implemented approach are:

Currently there are certain restrictions on parameters like speed of the vehicle, script on the number plate, skew in the image which can be aptly removed by enhancing the algorithms further.

Since the bounding boxes are rectangular in shape, slanting text is partially detected. The box doesn't bound the entire slant text. Hence some part of the text would be lost. The efficiency of the algorithms further reduce when the lighting conditions are improper.

FUTURE ENHANCEMENTS

Images with considerably more complex backgrounds must be tested in the future to enhance the efficiency. The text extraction on the system is performed by considering the transitions between edge of the text and background. Hence, the contrast must be clearly seen.

Approaches must be worked upon to detect skewed images and to improve the efficiency of the system when subjected to improper lighting conditions.

Recognition can be improved by introducing new features such as entropy values or continuous line segments. It can be further enhanced by introducing hierarchal classification methods while implementing SVM classifier.

BIBLIOGRAPHY

[1] Text Detection and Recognition in Imagery: A Survey, Qixiang Ye, IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume:37, Issue: 7), ISSN 0162-8828, IEEE 2015 July.

[2]Text Detection and Localization in Low Quality Video Images through Image Resolution Enhancement Technique, P. Rajendra Kumar

[3] A Design Flow for Robust License Plate Localization and Recognition in Complex Scenes Dhawal Wazalwar, Erdal Oruklu, Jafar Saniie, *Journal of Transportation Technologies*, 2012, 2, 13-21

[4] An Efficient Features – Based License Plate Localization Method, Hamid Mahini, ShohrehKasaei0-7695-2521-0/06/$20.00 (c) 2006 IEEE

[5] Automatic License Plate Localization Using Intrinsic Rules Saliency, Chirag N. Paunwala,Dr. Suprava Patnaik.

[6]A Design Flow for Robust License Plate Localization and Recognition in Complex Scenes

[7]X.Chen, A. Yuille, "Detecting and Reading Text in NaturalScenes",Computer Vision and Pattern Recognition (CVPR), pp. 366-373, 2004

[8] R. Lienhart, A. Wernicke, “Localizing and Segmenting Text in Images and Videos” Ieee Transactions On Circuits And Systems For Video Technology, Vol. 12, No. 4, April 2002, pp. 256-268

[9] A. Jain, B. Yu, “Automatic Text Location in Images and Video Frames”, Pattern Recognition 31(12): 2055-2076 (1998)

[10] H-K Kim, "Efficient automatic text location method and

content-based indexing and structuring of video database". J Vis Commun Image Represent 7(4):336–344 (1996)

[11] D. H. Ballard, C. M. Brown, Computer Vision, Pages 151 - 152, Prentice Hall, New Jersey, 1982.

[12] H. E. Kocer, K. K. Cevik, Artificial neural networks based vehicle license plate recognition, Procedia Computer Science, Volume 3, 2011, Pages 1033-1037

[13] F. Kahraman, B. Kurt,M. Gökmen, License Plate Character Segmentation Based on the Gabor Transform and Vector Quantization,ISCIS 2003,LNCS 2869,pp. 381 -388,2003.

[14] Y. Cui, Q. Huang, "Extracting characters of license plates from video sequences", Machine Vision and Applications, vol. 10, pp. 308-320, 1998.

[15] Y. R. Wang, W. H. Lin, S. J. Horng, A sliding window technique for efficient license localization based on discrete wavelet transform, Expert Systems with Applications, Volume 38, Issue 4, April 2011, Pages 3142-3146

[16] Sahana D. Gowda, P. Nagabhushan, Entropy Quantifiers useful for Establishing Equivalence between Text Document Images, International Conference on Computational Intelligence and Multimedia Applications, 2007. 0-7695-3050-8/07 IEEE.