***Project Report***

***on***

***Indian Number Plate Detection System***

Submitted in the partial fulfillment of the requirements for

the award of Degree of B. Tech

***By***

***Deepak Kumar Saini (1806810096)***

***Diksha(1806810106)***

***Deepanshu(1806810100)***

Under the Supervision of:-

Ms. Shweta Saxena

(Asst. Professor, Department of CSE)

****

Department of Computer Science ***&*** Engineering

Meerut Institute of Engineering and Technology,

Meerut – 250005



Dr. A.P.J. Abdul Kalam Technical University, U.P., Lucknow

[2018-2022]

**TABLE OF CONTENT** **Page No.**

**DECLARATION** **i**

**CERTIFICATE** **ii**

**ACKNOWLEDGEMENTS** **iii**

**ABSTRACT** **iv**

**CHAPTER 1INTRODUCTION**…..…………………………………………… 1

1.1 INTRODUCTION………...……………………………………………. 10

1.2 SCOPE…………………………………………………………………... 11

1.3 SOFTWARE DEVELOPMENT METHODOLOGY………………….. 12

**CHAPTER 2 LITERATURE SURVEY**………………………….. 14

2.1 EXISTING SYSTEM………………………………………..

2.1.1 EXISTING SYSTEM DISADVANTAGES………………..

2.2 PROBLEM STATEMENT………………………………….

2.3 PROPOSED SYSTEM…………………………………….

2.3.1 PROPOSED METHODOLOGIES………………………..

2.3.2 ADVANTAGES OF PROPOSED SYSTEM……………..

**CHAPTER 3 SOFTWARE REQUIREMENTS SPECIFICATION**………….. 17

3.1 INTRODUCTION………………………………………………………. 17

3.2 INTENDED AUDIENCE AND READING SUGGESTIONS………….. 17

3.3 GENERAL ARCHITECTURE OF SOFTWARE………………………. 18

3.4 REQUIREMENT SPECIFICATION………………………….................. 19

3.4.1 FUNCTIONAL REQUIREMENTS……………………………... 19

3.4.2 NON-FUNCTIONAL REQUIREMENTS………………………. 19

3.5 FEASIBILITY STUDY…………………………………………………. 20

3.5.1 OPERATIONAL FEASIBILITY………………………………..20

3.5.2 TECHNICAL FEASIBILITY…………………………………...20

3.4.3 ECONOMIC FEASIBILITY……………………………………20

3.6 SYSTEM REQUIREMENTS STUDY…………………………………… 21

3.6.1 SOFTWARE REQUIREMENTS……………………………….. 21

3.6.2 HARDWARE REQUIREMENTS……………………………... 21

3.7 USER REQUIREMENT DOCUMENT (URD)………………………….. 22

3.7.1 USE-CASE DIAGRAM………………………………………… 22

3.7.2 ACTIVITY DIAGRAM ………………………………………. … 23

3.8 SYSTEM DESIGN ……………………………………………………... 26

3.8.1 INTRODUCTION…………………………………………………. 26

3.8.2 DATA FLOW DIAGRAM ………………………………...... 26

|  |  |  |
| --- | --- | --- |
| 3.8.3 | SEQUENCE DIAGRAM……………………………………….. | 28 |
| 3.8.4 | CLASS DIAGRAM…………………………………………….. | 30 |
| **CHAPTER 4 SCREENSHOTS**………………………………………………… | | 31 |
| **CHAPTER 5 TECHNOLOGY USED**…………………………………………. | | 44 |
| 5.1 | PYTHON…………………………………………………………… | 44 |
| **CHAPTER 6 IMPLEMENTATION**………………………………   * 1. PLATFORM SELECTION………………………………………….   2. PROGRAMMING LANGUAGE GIST…………………………….   3. CODING STANDARDS……………………………………………   4. MODULE DESCRIPTION………………………………………….   **CHAPTER 7 TESTING AND INTEGRATION** ……………………………… | | 50 |
| 7.1. TEST CASE DESCRIPTION …………………………… | |  |
| 7.2. TYPES OF TESTING……………………………….. | | 50 |
| 7.3. TEST CASES ……………………………………… | | 53 |
| 7.4. FUTURE ENHANCEMENT …………………….. | | 56 |
| **CONCLUSION**.................................................................. | | 57 |
| **REFERENCES**………………………………………………………………. | | 58 |

**APPENDIX**……………………………………………………………….. A

***DECLARATION***

*I hereby declare that this submission is my work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material that to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.*

*Signature :*

*Name* *:Deepak Kumar Saini*

*Roll No. : 1806810096*

*Date* *:*

*Signature :*

*Name* *:Diksha*

*Roll No. : 1806810106*

*Date* *:*

*Signature :*

*Name* *:Deepanshu*

*Roll No. : 1806810100*

*Date* *:*

**CERTIFICATE**

This is to certify that *Project Report entitled ― Indian Number Plate Detection System* which is submitted by *Deepak Kumar Saini (1806810096),* in partial fulfillment of the requirement for the award of degree B. Tech. in Department of “*Computer Science and Engineering*” Of “*A.P.J. Abdul Kalam Technical University*”, is a record of the candidate own work carried out by him/her under my/our supervision. The matter embodied in this Project report is original and has not been submitted for the award of any other degree.

**Date:** **Supervisor**

**ACKNOWLEDGEMENTS**

*It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B.Tech Final Year. We owe a special debt of gratitude to our guide Ms. Shweta Saxena, Department of* “*Computer Science and Engineering*”*, “Meerut Institute of Engineering and Technology” for his constant support and guidance throughout our work. His sincerity, thoroughness, and perseverance have been a constant source of inspiration for us. It is only through his cognizant efforts that our endeavors have seen the light of the day.*

*We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution to the completion of the project.*

|  |  |  |
| --- | --- | --- |
| *Signature :* | *Signature:* | |
| *Name: Deepak Kumar Saini* | *:* | *Name:* | |
| *Roll No.: 1806810207* | *:* | *Roll No:* | |
| *Date:* | *:* | *Date:* | |

|  |
| --- |
| *Signature :* |
| *Name:* |
| *Roll No.:* |
| *Date* |

***ABSTRACT***

*Automatic Number Plate Recognition (ANPR) is a real time embedded system which identifies the characters directly from the image of the license plate. It is an active area of research. Vehicle number plate recognition (VNPR) has been intensively studied in many countries. Due to the different types of number plates being used, the requirements of an automatic number plate recognition system is different for each country. In this paper, a number plate localization and recognition system for vehicles in India is proposed. This system includes various operations such as taking pictures, localizing the number pad, truncating characters and OCR from alphanumeric characters. The main idea of this system is to design and develop effective image processing techniques and algorithms to localize the license plate in the captured image, to divide the characters from that number plate using segmentation and to identify each character of the segment by using the Open Computer Vision Library. This has been implemented in K-NN algorithm and python programming language. Many applications can be implemented by using this system, such as security, highway speed detection, violation of light, identification of handwritten text, discovery of stolen cars, automatic fee collection systems.*

CHAPTER 1

# Introduction

## INTRODUCTION

People from different countries interact in a multicultural environment to develop solutions to never-ending problems for men. The Open Source section is a one of the outstanding contribution in the scientific world is Python. Computer vision in the Intel’s research has been producing a fruit called Open Computer Vision (Open CV), which can support the development of computer vision [1]. At present, the use of vehicles is increasing throughout the country. All of these vehicles have a unique vehicle identification number as their main identifier. The ID is actually in the license number that refers to a legal license to participate in the public movement. Each vehicle in the world must have its own number plate that must be installed on its body (at least on the back). They need to Identify the vehicles are increasing in parallel with the number of vehicles. Vehicles in each country have a unique license number, which is written on its license plate. This number distinguishes one vehicle from the other, which is useful especially when both are of same make and model. An automated system can be implemented to identify the license plate of a vehicle and extract the characters from the region containing a license plate. The license plate number can be used to retrieve more information about the vehicle and its owner, which can be used for further processing. Such an automated system should be small in size, portable and be able to process data at sufficient rate.

This identification system helps with safety, automatic switching systems, highway speed detection, light detection, stolen vehicle detection, and human and non-human loss collection systems. The auto license plate recognizing system replaces the manual license plate number writing process in the computer system. In order to obtain an appropriate personal recognition, the license plate identification technique consists of three main topics. They are, find the location of the panel of digital images, segmentation the characters from the pictures of the panel and the visual character Recognition [2]. The most dominant and basic step is to determined the exact location of the number plate in the captured image. The localization of a license plate has been recognized either by structural analysis and color analysis method. In the License panel area, unwanted spots are removed by parsing the connected component. ANPR is a collective control system that captures the vehicle image and identifies the license number. Some ANPR system applications are automatic traffic control and tracking system, highway toll collection / automatic parking systems, petrol station automation, flight time monitoring. These systems automate the process of identifying vehicle license number, making it fast, cost effective.

## SCOPE

The main purpose of this project is to develop an efficient number plate recognition system by using the concept of image processing, which can be implemented various applications which includes vehicle tracking, traffic monitoring, automatic payment of tolls on highways or bridges, surveillance systems, tolls collection points, and parking management systems. The scope of this project is to detect the license plate from the given image and observe the output on monitor. This project can work as a base for future improvements in the field of image processing, especially in license plate extraction and plate number recognition

## SOFTWARE DEVELOPMENT METHODOLOGY

The flow chart of the proposed system is shown in Fig.1.The first step is vehicle number plate detection which is followed by character and number segmentation, feature extraction & recognition of the extracted feature and character recognition.

A license plate recognition system generally works in four main parts namely image acquisition, license plate detection, character segmentation and character recognition.

Fig.1.3

OCR

Character Segmentation

RGB to Gray scale

Extraction of Image/Video

Image/Video Analysis

# 2. CHAPTER 2 - LITERATURE SURVEY

[1] Vehicle body detection system utilizes the color characteristics of the barking lights to carry out detection. It first detects the location of the two barking lights in the captured image. Then set license plate detection region using the probability distribution of the license plate between the two lights, thus quickly locate the license plate. This method can eliminate any environmental interference during the license plate detection. From the results of experiment, it is determined that this system can effectively and quickly capture the vehicle image, detect and recognize the license plate whether it is dark, raining or under complicated environments.

[2] In this paper author proposed a system to localization of number plate mainly for the vehicles in West Bengal (India) and segmented the numbers as to identify each number separately. This presents an approach based on simple and efficient morphological operation and sobel edge detection method. Author also presents a simple approach to segmented all the letters and numbers used in the number plate. After reducing noise from the input image we try to enhance the contrast of the binarized image using histogram equalization. The system mainly concentrate on two steps; one is to locate the number plate and second is to segment all the number and letters to identify each number separately This method can eliminate any environmental interference during the license plate detection and improve the rate of accuracy of license plate detection and recognition.

[4] The system first senses the vehicle and then gets an image of vehicle from the front or back view of the vehicle. The system has four main steps to get the required information. These are image acquisition, plate localization, character segmentation and character recognition. This system is implemented and simulated in Python. The objective is to design an efficient automatic vehicle identification system by using the vehicle number plate, and to implement it for various applications such as automatic toll tax collection, parking system, Border crossings, Traffic control, stolen cars etc. The system has color image inputs of a vehicle and the output has the registration number of that vehicle. The system first senses the vehicle and then gets an image of vehicle from the front or back view of the vehicle. The system has four main steps to get the required information. These are image acquisition, plate localization, character segmentation and character recognition. This system is implemented and simulated in Python.

[5] Here Vehicle number plate is extracted by using the image segmentation and Optical character recognition technique which is used for the recognizing the character. And then resulting data is used to compare with the records on a database and data extracted from RFID. And in database there can be specific information like vehicle’s owner name, place of registration, or address, etc. The developed system firstly detects the vehicle RFID and then it captures the vehicle number plate. Here Vehicle number plate is extracted by using the image segmentation and Optical character recognition technique which is used for the recognizing the character. And then resulting data is used to compare with the records on a database and data extracted from RFID. And in database there can be specific information like vehicle’s owner name, place of registration, or address, etc. If the ID and the number are matches with the database then it show the message “authorized person” else “unauthorized person”. Both should be match with the database. If single one condition is true then it shows “unauthorized person”. Here the system also adding the advantage of identifying and auto information about theft and crime vehicles.

**2.1 EXISTING SYSTEM**:

The car plates appear in different types of character styles, either single or double row, different sizes, spacing and character counts. Due to such kind of variations even localizing or detecting these plates becomes a tedious process. In the existing system, foreground estimation is done by Gaussian mixture model then proposing a real time and robust method of license plate extraction based on block variance technique. License plate extraction is an important stage in license plate recognition for automated transport system. The Extracted license plates are segmented into individual characters by using a regionbased approach. The recognition scheme combines adaptive iterative thresholding with a template matching algorithm.

**2.1.1 EXISTING SYSTEM DISADVANTAGES**

• The existing method cannot work properly on degraded images with a complex background.

• Low accuracy.

• Noise content was high.

• They do not take into consideration of the noise or the image normalization in the input image.

• These methods do not show high contrast image for the output image.

• Manual assessment is subjective, time consuming and expensive.

• In these methods, selection of features and classification strategy is difficult and challenging.

• Poor and Inaccurate segmentation result.

**2.2 PROBLEM STATEMENT**

First, it is necessary to locate and extract the license plate region from a larger scene image. Second, having a license plate region to work with, the alphanumeric characters in the plate need to be extracted from the background. Third, deliver them to an OCR system for recognition. In order to identify a vehicle by reading its license plate successfully, it is obviously necessary to locate the plate in the scene image provided by some acquisition system (e.g. video or still camera). Locating the region of interest helps in dramatically reducing both the computational expense and algorithm complexity. For example, a currently common 1024x768 resolution image contains a total of 786,432 pixels, while the region of interest (in this case a license plate) may account for only 10% of the image area. Also, the input to the following segmentation and recognition stages should be simplified, resulting in easier algorithm design and shorter computation times.

**2.3 PROPOSED SYSTEM**

This project is on the development of new approaches for extraction of license plates. The proposed algorithm is based on video acquisition, extraction of plate region, segmentation of plate characters and recognition of characters. Extraction of plate is a difficult task. In this project, a simple license plate extraction method is presented. The method is basically based on the Edge Detection algorithm including four major stages, which are RGB to gray-scale conversion, Gaussian Blurring, morphological operations and extracting the accurate location of the license plate. Mean squared error method is used for recognition of characters.

**2.3.1 PROPOSED METHODOLOGIES**

•Image/Video acquisition

• RGB to gray scale conversion

• Sobel edge detection

• Thresholding

• Cluster based segmentation

• Localization

• Character segmentation

• Character recognition using OCR

**2.3.2 ADVANTAGES OF PROPOSED SYSTEM**

• Our proposed method shows better performance compared to existing.

• It is simple, robust, and involves minimum parameter tuning.

• Accurate segmentation result of license plate characters.

• The noise content is removed.

**2.4 OBJECTIVES**

The main objective in this research project is to experiment deeply and find alternative solutions to the image segmentation and character recognition problems within the License Plate Recognition framework. To develop a system in python which can perform detection as well as recognition of car number plate.

•Find a method with acceptable results for the correct location of the area of the license plate.

• Build a system that determines the characters of the license plate that is localized from a video frame.

• Recognize each character we have extracted above by mean squared error method.

# CHAPTER 3

# SOFTWARE REQUIREMENTS SPECIFICATION

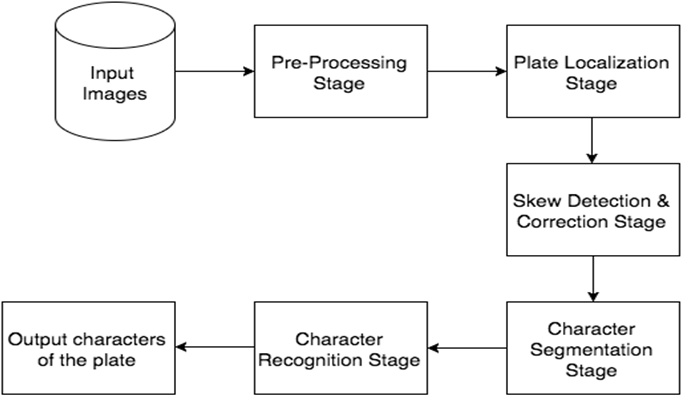
## 3.1 INTRODUCTION

Software Requirements Specification present a detailed description of the Vehicle Number Plate Detection System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli.

## 3.2 INTENDED AUDIENCE AND READING SUGGESTIONS

The document is intended for requirements engineer, domain expert, developers, project manager and stakeholders.

## 3.3 GENERAL ARCHITECTURE OF SOFTWARE



*Fig 3.3.1* – General tier Architecture

## REQUIREMENT SPECIFICATION

### FUNCTIONAL REQUIREMENTS

Functional requirement refers to the functionalities that are applicable to a system. The functional requirements of automatic license plate recognition system are stated below. The system must be able to:

1. Load videos from the system.

2. Extract frames from the video.

3. Localize license plate region from the frames.

4. Segment characters from the localized plate.

5. Recognize the segmented characters and display it on the terminal.

### NON-FUNCTIONAL REQUIREMENTS

A non-functional requirement is a system must behave or how is the system's behavior. This also specifies how the system's quality characteristics or quality attributes. In order to put this constraint upon the specific system behavior, the qualities goals of the designed system should go in these:

Execution qualities:

• Functionality

• Security

• Usability

• Effectiveness & Efficiency

Evolution qualities:

• Availability

• Reliability

• Manageability

## USER REQUIREMENTS

1. The ANPR system should captures the videos of vehicle number plates from moving vehicles in both day and night time conditions.

2. Quick detection of the vehicles’ number plates.

3. Online optical character recognition (OCR) of the captured vehicles’ number plates.

4. Providing an operator control of the observed by the ANPR system object, including the supervising the ANPR live video-stream of the controlled road, lane, etc. and visualization of the found number plates

## SYSTEM REQUIREMENTS STUDY

### HARDWARE REQUIREMENTS

The following is the suggested minimum system configuration to run the ANPR software:

• Intel i3 CPU or higher

• Dual core

• 1 TB HARD DISK

• 2GB RAM or more

• Windows 7 (64 bit), Windows 8 (64 bit), Windows 10 (64 bit)

### SOFTWARE REQUIREMENTS

• PYTHON VERSION 3.6

Python is a high level, interpreted and general purpose dynamic programming language that focuses on code readability. It has fewer steps when compared to Java and C. It was founded in 1991 by developer Guido Van Rossum. It is used in many organizations as it supports multiple programming paradigms. It also performs automatic memory management.

Advantages of Python

The diverse application of the Python language is a result of the combination of features which give this language an edge over others. Some of the benefits of programming in Python include

1. Presence of Third Party Modules

The Python Package Index (PyPI) contains numerous third-party modules that make Python capable of interacting with most of the other languages and platforms.

1. Extensive Support Libraries

Python provides a large standard library which includes areas like internet protocols, string operations, web services tools and operating system interfaces. Many high use programming tasks have already been scripted into the standard library which reduces length of code to be written significantly.

1. Open Source and Community Development

Python language is developed under an OSI-approved open source license, which makes it free to use and distribute, including for commercial purposes. Further, its development is driven by the community which collaborates for its code through hosting conferences and mailing lists, and provides for its numerous modules.

1. Learning Ease and Support Available

Python offers excellent readability and uncluttered simple-to-learn syntax which helps beginners to utilize this programming language. The code style guidelines, PEP 8, provide a set of rules to facilitate the formatting of code. Additionally, the wide base of users and active developers has resulted in a rich internet resource bank to encourage development and the continued adoption of the language.

1. User– friendly Data Structures

Python has built-in list and dictionary data structures which can be used to construct fast runtime data structures. Further, Python also provides the option of dynamic high-level data typing which reduces the length of support code that is needed.

1. Productivity and Speed

Python has clean object-oriented design, provides enhanced process control capabilities, and possesses strong integration and text processing capabilities and its own unit testing framework, all of which contribute to the increase in its speed and productivity. Python is considered a viable option for building complex multi-protocol network applications. As can be seen from the above-mentioned points, Python offers a number of advantages for software development. As upgrading of the language continues, its loyalist base could grow as well.

• Python File

The standard Python installer already associates the .py extension with a file type (Python.File) and gives that file type an open command that runs the interpreter. This is enough to make scripts executable from the command prompt as ‘foo.py’. If you’d rather be able to execute the script by simple typing ‘foo’ with no extension you need to add .py to the PATHEXT environment variable.

• Creating and Running Script File

To create scripts files, you need to use a text editor. You can open the python editor in two ways:

•Using the command prompt

•Using the IDE

If you are using the command prompt, type edit in the command prompt. This will open the editor. You can directly type edit and then the filename (with .py extension).

## USER REQUIREMENT DOCUMENT (URD)

### USE-CASE DIAGRAM

Actor:

• User

Use Case:

• Capture video and acquire image

• Verify Vehicle

• Identify Number

Precondition:

• A camera is placed at 4-5 m away from the vehicle to get the clear view of the number plate.

• Videos are captured and stored in a repository.

Post condition:

• The license plate numbers are recognized and displayed on the terminal.



Fig 3.7.1.1- Use Cases and Actors

### ACTIVITY DIAGRAM

Description:

The main aim behind the number plate recognition is the storage of information of vehicles with respect to its number plate characters. The information thus can be used to track the vehicles.

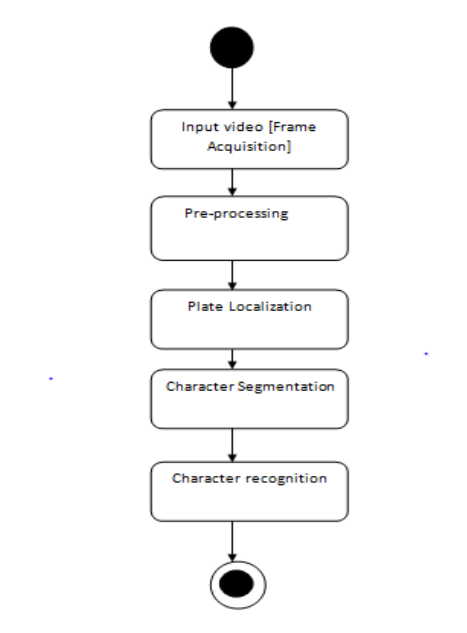


Fig 3.7.2.1 – Activity Diagram

## SYSTEM DESIGN

### INTRODUCTION

Systems design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization.

UML is used as a modeling language to express design aspects of the system. Unified Modeling Language (UML) describes software both structurally and behaviorally with graphical notation.

System architecture is the conceptual model that defines the structure, behaviour and views of a system. The below figure is an architectural design for the Automatic Number Plate Recognition (ANPR) system. ANPR system is a system that reads and process video that consists of vehicle number plate as input and recognizes the number plate as output automatically.

### DATA FLOW DIAGRAM

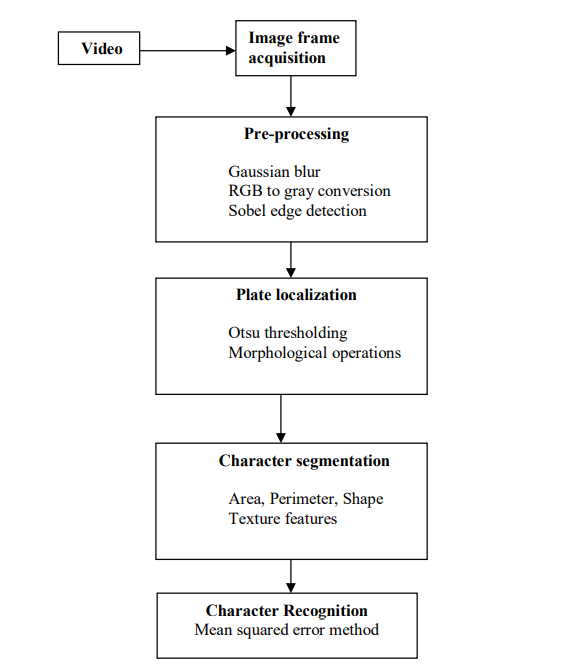


Fig 3.8.2.1- Data Flow Diagram

### SEQUENCE DIAGRAM

**Basic Flow:**

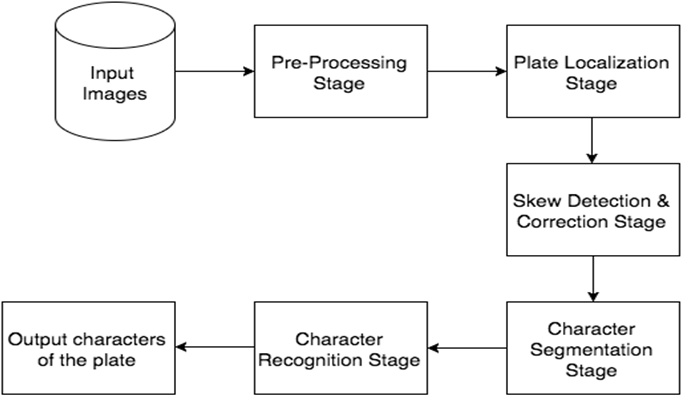


Fig 3.8.3.1 – Basic Flow Diagram

### CLASS DIAGRAM

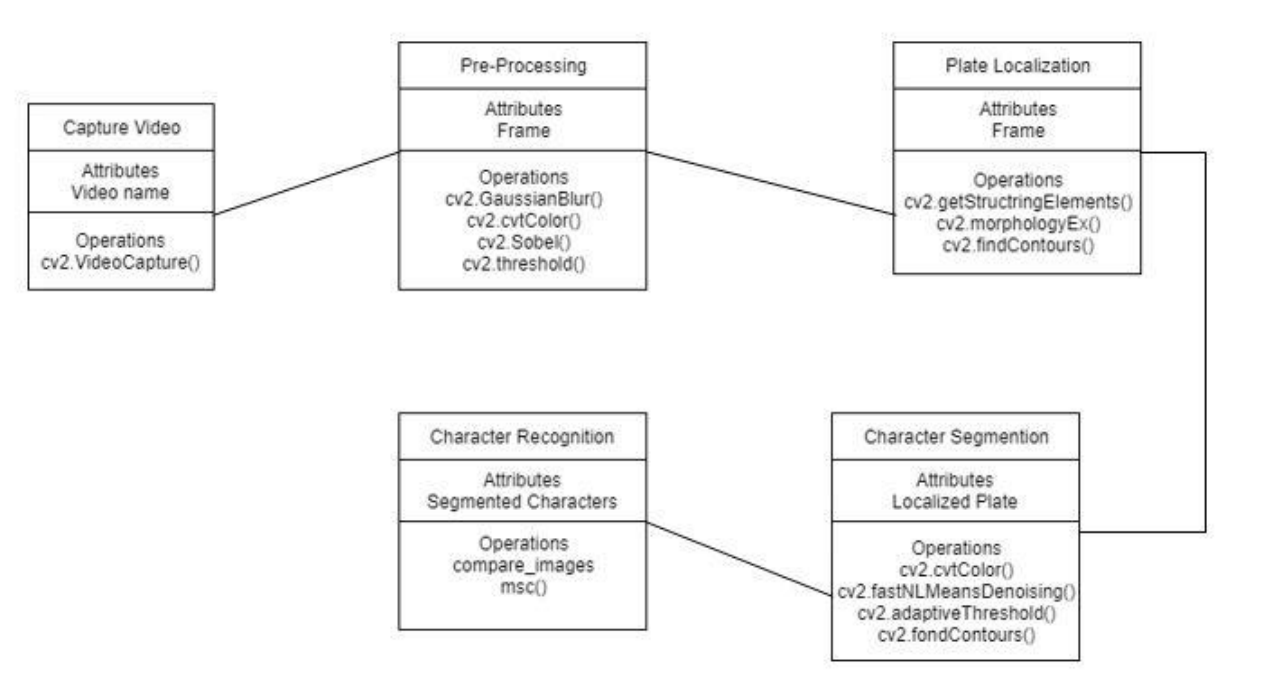


Fig 3.8.4.1- Class Model

# CHAPTER 4 SCREENSHOTS



Fig 4.1 –Input Video

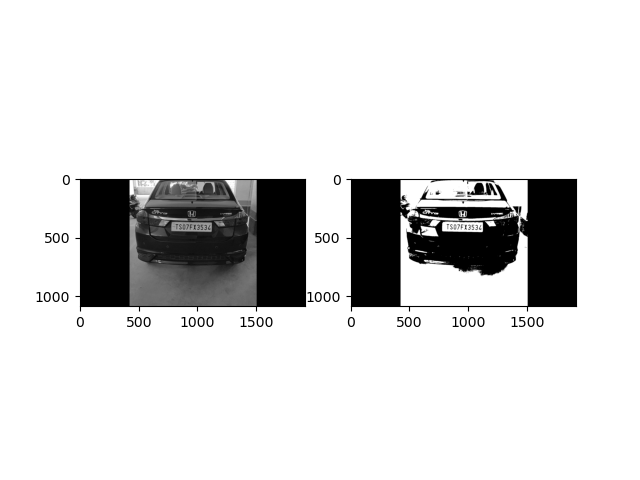


Fig 4.2

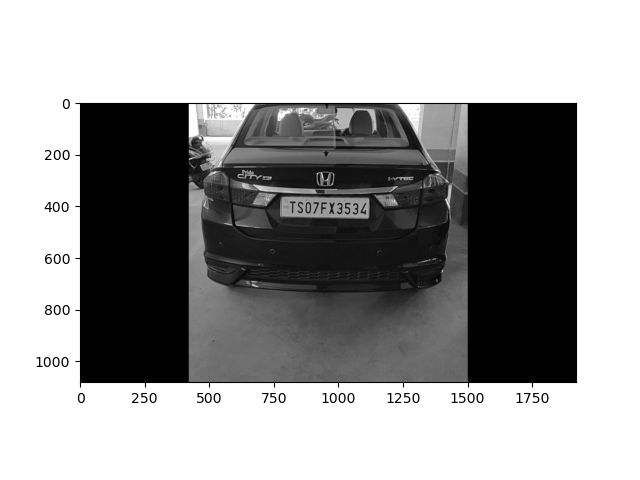


Fig 4.3 Gray scale image

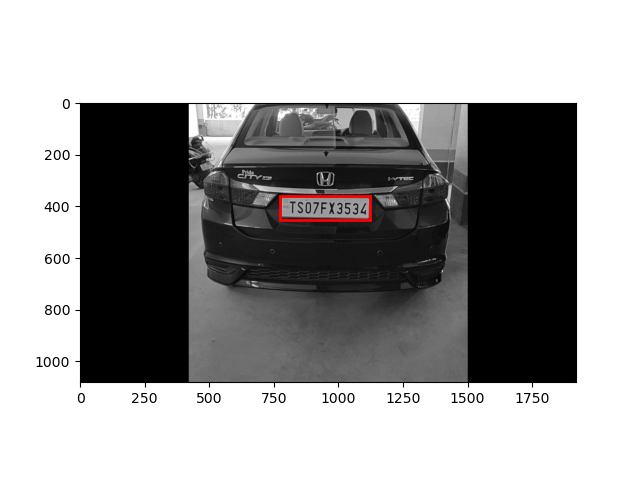


Fig 4.4 Number Plate Extraction

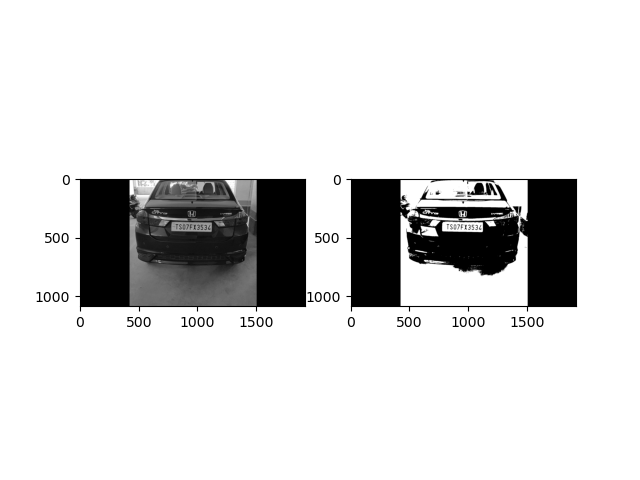


Fig 4.5

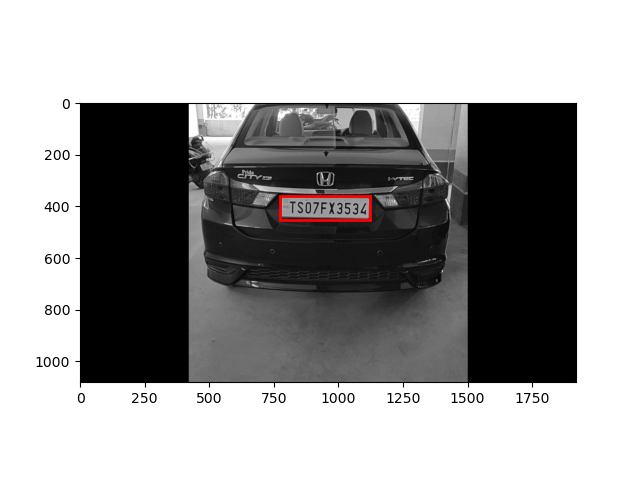


Fig 4.6

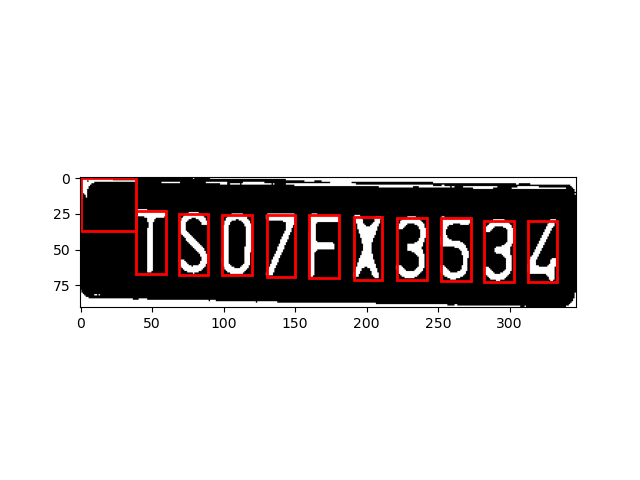


Fig 4.7 Segementation of characters

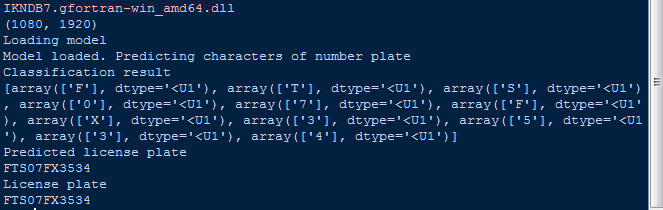


Fig 4.8 Final Output

# CHAPTER 5 TECHNOLOGY USED

## Python

Python is an interpreted, interactive, object-oriented programming language. Itincorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. It supports multiple programming paradigms beyond object-oriented programming, such as procedural and functional programming. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface. Finally, Python is portable: it runs on many Unix variants including Linux and macOS, and on Windows.

The Python Software Foundation is an independent non-profit organization that holds the copyright on Python versions 2.1 and newer. The PSF’s mission is to advance open source technology related to the Python programming language and to publicize the use of Python.

You can do anything you want with the source, as long as you leave the copyrights in and display those copyrights in any documentation about Python that you produce. If you honor the copyright rules, it’s OK to use Python for commercial use, to sell copies of Python in source or binary form (modified or unmodified), or to sell products that incorporate Python in some form. We would still like to know about all commercial use of Python, of course.

Python is a high-level general-purpose programming language that can be applied to many different classes of problems.

The language comes with a large standard library that covers areas such as string processing (regular expressions, Unicode, calculating differences between files), internet protocols (HTTP, FTP, SMTP, XML-RPC, POP, IMAP, CGI programming), software engineering (unit testing, logging, profiling, parsing Python code), and operating system interfaces (system calls, filesystems, TCP/IP sockets). Look at the table of contents for the Python Standard Library to get an idea of what’s available. A wide variety of third-party extensions are also available. Consult the Python Package Index to find packages of interest to you.

# chapter 6 Implementation

6.1 PLATFORM SELECTION

IDLE (short for integrated development environment or integrated development and learning environment) is an integrated development environment for Python, which has been bundled with the default implementation of the language since 1.5.2b1. It is packaged as an optional part of the Python packaging with many Linux distributions. It is completely written in Python and the Tkinter GUI toolkit (wrapper functions for Tcl/Tk).

IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment. To that end, it is cross-platform, and avoids feature clutter.

According to the included README, its main features are:

• Multi-window text editor with syntax highlighting, autocompletion, smart indent and other.

• Python shell with syntax highlighting.

• Integrated debugger with stepping, persistent breakpoints, and call stack visibility.

IDLE has been criticized for various usability issues, including losing focus, lack of copying to clipboard feature, lack of line numbering options, and general user interface design; it has been called a "disposable" IDE, because users frequently move on to a more advanced IDE as they gain experience.

* 1. PROGRAMMING LANGUAGE GIST

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

* 1. CODING STANDARDS

1. Naming Conventions

Module Names: – Short, lowercase names, without underscores. Example: myfile.py Class Names: – CapWords convention. Example: MyClass Exception Names: – If a module defines a single exception raised for all sorts of conditions, it is generally called "Error". Otherwise use CapWords convention (i.e. MyError.)1. Method Names and Instance Variables: – The “Style Guide for Python Code” recommends using lowercase with words separated by underscores (example: my\_variable). But since most of the code uses mixedCase, recommend using this style (example: myVariable ) – Use one leading underscore only for internal methods and instance variables (i.e. protected). Example: \_myProtectedVar – Use two leading underscores to denote class-private names Example: \_\_myPrivateVar – Don’t use leading or trailing underscores for public attributes unless they conflict with reserved words, in which case, a single trailing underscore is preferrable (example: class \_ ) .

1. Organizing imports

They should be always put at the top of the file, just after any module comments and document strings, and before module global and constants. Imports should be on separate lines.

Wrong: import sys, os

Right: import sys import os

The following is OK, though: from types import String Type, List Type Imports should be grouped in the following order with a blank line between each group of imports: – standard library imports – related major package imports – application specific imports.

1. Indentation and Line lengths

Indentions: – 2 spaces (no tabs!) – Avoid using more than five levels of indention. Line length: – Maximum of 72 characters (never exceed 79 characters) – You can break a long line using “\”.

1. Break Lines

Leave one line between functions in a class. Extra blank lines may be used to separate groups of related functions. Blank lines may be omitted between a bunch of related oneliners. Use blank lines in functions, sparingly, to indicate logical sections.

1. White Space

Multiple statements on the same line are discouraged.

WRONG: if foo == 'blah': doBlahThing()

CORRECT: if foo == 'blah': doBlahThing()

No white space immediately before an open parenthesis.

WRONG: spam (1)

CORRECT: spam(1)

WRONG: dict ['key'] = list [index]

CORRECT: dict['key'] = list[index]

No white space inside parentheses, brackets or braces.

WRONG: spam( ham[ 1 ], { eggs: 2 } )

CORRECT: spam(ham[1], {eggs:2}) ƒ

No white space immediately before a comma, semicolon, or colon.

WRONG: if x == 4 : print x , y ; x , y = y , x

CORRECT: if x == 4: print x, y; x, y = y, x 17 White Space (cont.) ƒ

No more than one space around an operator.

WRONG: x = 1 yVal = 2 longVariable = 3

CORRECT: x = 1 yVal = 2 longVariable = 3 18 White Space (cont.) ƒ

Always surround the following operators with a single space on either side – assignment

(=) – comparisons (==, , !=, <>, <=, >=, in, not in, is, is not) – Booleans (and, or, not) –

Arithmetic operators (+, -, \*, /, %)

WRONG: if (x==4)or(x==5): x=y+5

CORRECT: if (x == 4) or (x == 5): x = y + 5 19 White Space (cont.) ƒ

Don't use spaces around the '=' sign when used to indicate a keyword argument or a

default parameter value.

WRONG: def complex(real, imag = 0.0): return magic(r = real, i = imag)

CORRECT: def complex(real, imag=0.0): return magic(r=real, i=imag)

1. Comments

Block Comments: They are indented to the same level as the code they apply to. Each line of a block comment starts with a # and a single space. Paragraphs inside a block comment are separated by a line containing a single #. Block comments are best surrounded by a blank line above and below them Example: # Compensate for border. This is done by incrementing x # by the same amount x += 1 25

Inline Comments: They should start with a # and a single space. Should be separated by at least two spaces from the statement they apply to. Example: x += 1 # Compensate for border.

1. Document Strings

Write document strings for all public modules, functions, classes, and methods. Document strings are not necessary for nonpublic methods, but you should have a comment that describes what the method does. This comment should appear after the "def" line. Insert a blank line before and after all document strings that document a class.

One-line Document strings: The opening and closing """ are on the same line. – There is no blank line either before or after the document string. Describes the function or method's effect as a command ("Do this", "Return that"), not as a description.

Multi-line Document strings: – The """ that ends a multiline document string should be on a line by itself. – Script: The document string of a script should be usable as its "usage" message. It should document the script's function, the command line syntax, and the environment variables. – Module: The document string for a module should generally list the classes, exceptions and functions (and any other objects) that are exported by the module, with a one-line summary of each.

Class: The document string for a class should summarize its behavior and list the public methods and instance variables. If the class is intended to be sub classed, and has an additional interface for subclasses, this interface should be listed separately. If a class subclasses another class and its behavior is mostly inherited from that class, its document string should mention this and summarize the differences. The class constructor should be documented in the document string for its \_\_init\_\_ method.

Function or method: The document string should summarizes its behavior and document its arguments, return value, side effects, exceptions raised, and restrictions on when it can be called. Optional arguments should be indicated. Use the verb "override" to indicate that a subclass method replaces a super class method and does not call the super class method; use the verb "extend" to indicate that a subclass method calls the super class method. The document string should contain a summary line, followed by a blank line, followed by a more elaborate description.

1. Library used
2. OpenCV-python

cv2 is the module import name for opencv-python.

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. It can process images and videos to identify objects, faces, or even the handwriting of a human.

OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development.

OpenCV-Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language. OpenCV-Python is a library of Python bindings designed to solve computer vision problems. OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

2. NumPy

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use  it freely.

NumPy stands for Numerical Python and it is denoted as np.

NumPy is a general-purpose array-processing package. It provides a high-performance  multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features  including these important ones:

A powerful N-dimensional array object .

Sophisticated (broadcasting) functions.

Tools for integrating C/C++ and Fortran code.

Useful linear algebra, Fourier transform, and random number capabilities Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional  container of generic data.

Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and  speedily integrate with a wide variety of databases.

3. cycler

The public API of **cycler** consists of a class [**Cycler**](https://matplotlib.org/cycler/generated/cycler.Cycler.html#cycler.Cycler), a factory function [**cycler()**](https://matplotlib.org/cycler/generated/cycler.cycler.html#cycler.cycler), and a concatenation function [**concat()**](https://matplotlib.org/cycler/generated/cycler.concat.html#cycler.concat). The factory function provides a simple interface for creating ‘base’ [**Cycler**](https://matplotlib.org/cycler/generated/cycler.Cycler.html#cycler.Cycler) objects while the class takes care of the composition and iteration logic.

4. SKLEARN

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in python. it  provides a selection of efficient tools for machine learning and statistical modelling including  classification, regression, clustering and dimensionality reduction via a consistence interface  in python. this library, which is largely written in python, is built upon Numpy, SciPy and  Matplotlib.

5. MATPLOTLIB

Matplotlib is a low-level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it  freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C  and JavaScript for Platform compatibility. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

6. imutils

Imutils is a package based on OpenCV, which can call the opencv interface more simply. It can easily realize a series of operations such as image translation, rotation, scaling, skeletonization and so on.

Installation method: pip install imutils

* 1. Module Description

The proposed system has the following four modules:

* + 1. PREPROCESSING

Pre-processing steps involves the following methods:

1. Gaussian blur: Gaussian blurring is highly effective in removing gaussian noise from the image. Image blurring is achieved by convolving the image with a low-pass filter kernel. It actually removes high frequency content (e.g: noise, edges) from the image resulting in edges being blurred. In this approach, instead of a box filter consisting of equal filter coefficients, a Gaussian kernel is used. It is done with the function, cv2.GaussianBlur().



Fig 6.4.1.1 Input Image

1. Gray scale conversion: It involves conversion of RGB image into grey image. A gray-scale image is composed of different shades of grey color. A true color image can be converted to a gray scale image by preserving the luminance (brightness) of the image. Here the RGB image is a combination of RED, BLUE AND GREEN colors. The grayscale image is obtained from the RGB image by combining 30% of RED, 60% of GREEN and 11% of BLUE. This gives the brightness information of the image. The resulting image will be two dimensional. The value 0 represents black and the value 255 represents white. The range will be between black and white values.

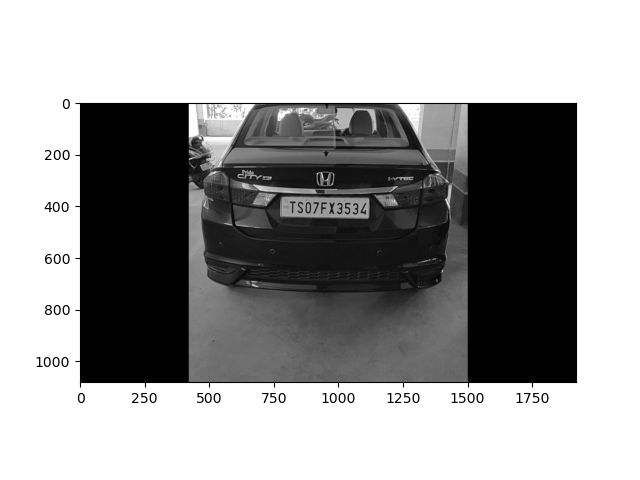
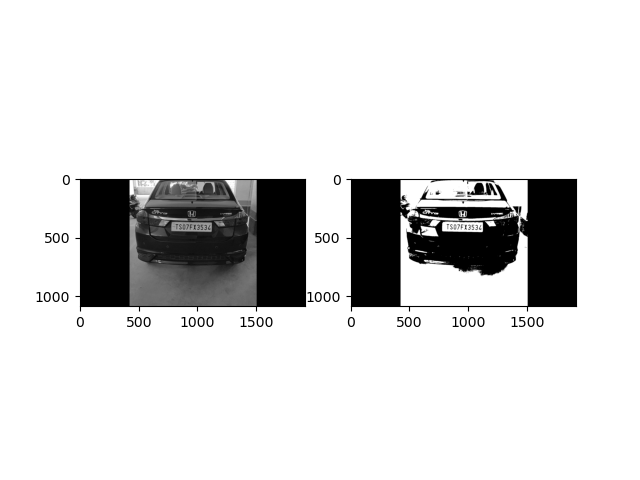


Fig 6.4.1.2 Gray Scale Image

1. Sobel operator: It is used in image processing and computer vision, particularly within edge detection algorithms where it creates an image emphasizing edges. The Sobel edge detector is a gradient based method. It works with first order derivatives. It calculates the first derivatives of the image separately for the X and Y axes. The derivatives are only approximations (because the images are not continuous).

Fig 6.4.1.3 Sobel Image

1. Thresholding: Converts gray scale image to binary image. If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). The function used is cv2.threshold. First argument is the source image, which should be a grayscale image.

Otsu thresholding

One of the most used techniques for the analysis of the images is thresholding, i.e, the application of a threshold along a particular scale of values, to filter in some way an image. It converts any image in grayscale (or color) into a totally black and white image. Often this is very useful for recognizing the regular shapes, contours within an image, or even to delimit and divide zones inside, to then be used in a different way in the subsequent processing.

cv2.threshold(img, im\_bw, 0,255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

• img is a source 8-bit image,

• im\_bw is the result,

• 0 means threshold level which actually is omitted because we usedcv2.THRESH\_OTSU flag,

• 255 is a value that is going to be assigned to pixels in the result (namely, to all pixels whose value in the source is greater than computed threshold level)

• cv2.THRESH\_BINARY + cv2.THRESH\_OTSU is a required flag to perform Otsu• thresholding. Because in fact we would like to perform binary thresholding, so we use CV\_THRESH\_BINARY combined with CV\_THRESH\_OTSU

PSEUDO CODE for #DetectPlate

from skimage.io import imread

from skimage.filters import threshold\_otsu

import matplotlib.pyplot as plt

import imutils

import cv2

from skimage import measure

from skimage.measure import regionprops

import matplotlib.pyplot as plt

import matplotlib.patches as patches

import os

import shutil

filename = 'e:/project/video12.mp4'

if os.path.exists('output'):

shutil.rmtree('output')

os.makedirs('output')

cap = cv2.VideoCapture(filename)

# cap = cv2.VideoCapture(0)

count = 0

while cap.isOpened():

ret,frame = cap.read()

if ret == True:

cv2.imshow('window-name',frame)

cv2.imwrite("./output/frame%d.jpg" % count, frame)

count = count + 1

if cv2.waitKey(10) & 0xFF == ord('q'):

break

else:

break

cap.release()

cv2.destroyAllWindows()

# car image -> grayscale image -> binary image

car\_image = imread("./output/frame%d.jpg"%(count-1), as\_gray=True)

car\_image = imutils.rotate(car\_image, 270)

# car\_image = imread("car.png", as\_gray=True)

# it should be a 2 dimensional array

print(car\_image.shape)

# the next line is not compulsory however, a grey scale pixel

# in skimage ranges between 0 & 1. multiplying it with 255

# will make it range between 0 & 255 (something we can relate better with

gray\_car\_image = car\_image \* 255

fig, (ax1, ax2) = plt.subplots(1, 2)

ax1.imshow(gray\_car\_image, cmap="gray")

threshold\_value = threshold\_otsu(gray\_car\_image)

binary\_car\_image = gray\_car\_image > threshold\_value

# print(binary\_car\_image)

ax2.imshow(binary\_car\_image, cmap="gray")

# ax2.imshow(gray\_car\_image, cmap="gray")

plt.show()

# CCA (finding connected regions) of binary image

# this gets all the connected regions and groups them together

label\_image = measure.label(binary\_car\_image)

# print(label\_image.shape[0]) #width of car img

# getting the maximum width, height and minimum width and height that a license plate can be

plate\_dimensions = (0.03\*label\_image.shape[0], 0.08\*label\_image.shape[0], 0.15\*label\_image.shape[1], 0.3\*label\_image.shape[1])

plate\_dimensions2 = (0.08\*label\_image.shape[0], 0.2\*label\_image.shape[0], 0.15\*label\_image.shape[1], 0.4\*label\_image.shape[1])

min\_height, max\_height, min\_width, max\_width = plate\_dimensions

plate\_objects\_cordinates = []

plate\_like\_objects = []

fig, (ax1) = plt.subplots(1)

ax1.imshow(gray\_car\_image, cmap="gray")

flag =0

# regionprops creates a list of properties of all the labelled regions

for region in regionprops(label\_image):

# print(region)

if region.area < 50:

#if the region is so small then it's likely not a license plate

continue

# the bounding box coordinates

min\_row, min\_col, max\_row, max\_col = region.bbox

# print(min\_row)

# print(min\_col)

# print(max\_row)

# print(max\_col)

region\_height = max\_row - min\_row

region\_width = max\_col - min\_col

# print(region\_height)

# print(region\_width)

# ensuring that the region identified satisfies the condition of a typical license plate

if region\_height >= min\_height and region\_height <= max\_height and region\_width >= min\_width and region\_width <= max\_width and region\_width > region\_height:

flag = 1

plate\_like\_objects.append(binary\_car\_image[min\_row:max\_row,

min\_col:max\_col])

plate\_objects\_cordinates.append((min\_row, min\_col,

max\_row, max\_col))

rectBorder = patches.Rectangle((min\_col, min\_row), max\_col - min\_col, max\_row - min\_row, edgecolor="red",

linewidth=2, fill=False)

ax1.add\_patch(rectBorder)

# let's draw a red rectangle over those regions

if(flag == 1):

# print(plate\_like\_objects[0])

plt.show()

if(flag==0):

min\_height, max\_height, min\_width, max\_width = plate\_dimensions2

plate\_objects\_cordinates = []

plate\_like\_objects = []

fig, (ax1) = plt.subplots(1)

ax1.imshow(gray\_car\_image, cmap="gray")

# regionprops creates a list of properties of all the labelled regions

for region in regionprops(label\_image):

if region.area < 50:

#if the region is so small then it's likely not a license plate

continue

# the bounding box coordinates

min\_row, min\_col, max\_row, max\_col = region.bbox

# print(min\_row)

# print(min\_col)

# print(max\_row)

# print(max\_col)

region\_height = max\_row - min\_row

region\_width = max\_col - min\_col

# print(region\_height)

# print(region\_width)

# ensuring that the region identified satisfies the condition of a typical license plate

if region\_height >= min\_height and region\_height <= max\_height and region\_width >= min\_width and region\_width <= max\_width and region\_width > region\_height:

# print("hello")

plate\_like\_objects.append(binary\_car\_image[min\_row:max\_row,

min\_col:max\_col])

plate\_objects\_cordinates.append((min\_row, min\_col,

max\_row, max\_col))

rectBorder = patches.Rectangle((min\_col, min\_row), max\_col - min\_col, max\_row - min\_row, edgecolor="red",

linewidth=2, fill=False)

ax1.add\_patch(rectBorder)

# let's draw a red rectangle over those regions

# print(plate\_like\_objects[0])

plt.show()

* + - 1. PLATE LOCALISATION

Plate localization is based on morphological operations and aspect ratio.

1. Morphological operation

Binary images may contain numerous imperfections. In particular, the binary regions produced by simple thresholding are distorted by noise and texture. Morphological image processing pursues the goals of removing these imperfections by accounting for the form and structure of the image. These techniques can be extended to grayscale images. Morphological techniques probe an image with a small shape or template called a structuring element. The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighborhood of pixels. Some operations test whether the element "fits" within the neighborhood, while others test whether it "hits" or intersects the neighborhood. A morphological operation on a binary image creates a new binary image in which the pixel has a non-zero value only if the test is successful at that location in the input image. When a structuring element is placed in a binary image, each of its pixels is associated with the corresponding pixel of the neighborhood under the structuring element. The structuring element is said to fit the image if, for each of its pixels set to 1, the corresponding image pixel is also 1. Similarly, a structuring element is said to hit, or intersect, an image if, at least for one of its pixels set to 1 the corresponding image pixel is also 1.

1. Localization:

Based on the aspect ratio of the contours, license plate candidate is selected.

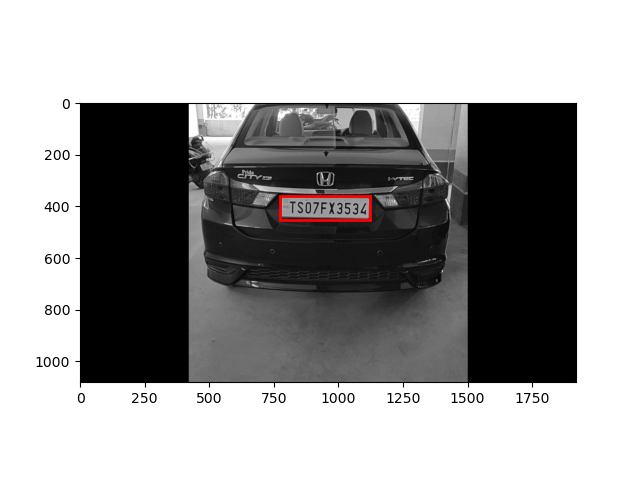


Fig 6.4.2.1 Localization of plate

* + 1. CHARACTER SEGMENTATION

Code to segment individual characters of the license plate

import numpy as np

from skimage.transform import resize

from skimage import measure

from skimage.measure import regionprops

import matplotlib.patches as patches

import matplotlib.pyplot as plt

import DetectPlate

# The invert was done so as to convert the black pixel to white pixel and vice versa

license\_plate = np.invert(DetectPlate.plate\_like\_objects[0])

labelled\_plate = measure.label(license\_plate)

fig, ax1 = plt.subplots(1)

ax1.imshow(license\_plate, cmap="gray")

# the next two lines is based on the assumptions that the width of

# a license plate should be between 5% and 15% of the license plate,

# and height should be between 35% and 60%

# this will eliminate some

character\_dimensions = (0.35\*license\_plate.shape[0], 0.60\*license\_plate.shape[0], 0.05\*license\_plate.shape[1], 0.15\*license\_plate.shape[1])

min\_height, max\_height, min\_width, max\_width = character\_dimensions

characters = []

counter=0

column\_list = []

for regions in regionprops(labelled\_plate):

y0, x0, y1, x1 = regions.bbox

region\_height = y1 - y0

region\_width = x1 - x0

if region\_height > min\_height and region\_height < max\_height and region\_width > min\_width and region\_width < max\_width:

roi = license\_plate[y0:y1, x0:x1]

# draw a red bordered rectangle over the character.

rect\_border = patches.Rectangle((x0, y0), x1 - x0, y1 - y0, edgecolor="red", linewidth=2, fill=False)

ax1.add\_patch(rect\_border)

# resize the characters to 20X20 and then append each character into the characters list

resized\_char = resize(roi, (20, 20))

characters.append(resized\_char)

# this is just to keep track of the arrangement of the characters

column\_list.append(x0)

# print(characters)

plt.show()

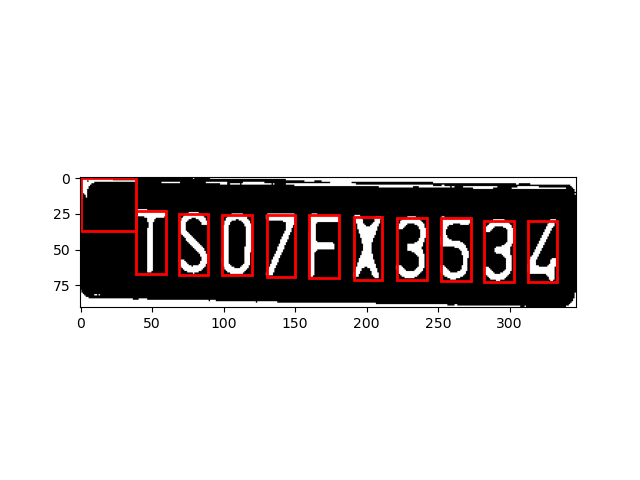


Fig 6.4.3.1 Segmented Characters

* + 1. TRAIN RECOGNIZE CHARACTERS

#Code\_for\_training\_the\_model

import os

import numpy as np

from sklearn.svm import SVC

from sklearn.model\_selection import cross\_val\_score

from skimage.io import imread

from skimage.filters import threshold\_otsu

letters = [

'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D',

'E', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'P', 'Q', 'R', 'S', 'T',

'U', 'V', 'W', 'X', 'Y', 'Z'

]

def read\_training\_data(training\_directory):

image\_data = []

target\_data = []

for each\_letter in letters:

for each in range(10):

image\_path = os.path.join(training\_directory, each\_letter, each\_letter + '\_' + str(each) + '.jpg')

# read each image of each character

img\_details = imread(image\_path, as\_gray=True)

# converts each character image to binary image

binary\_image = img\_details < threshold\_otsu(img\_details)

# the 2D array of each image is flattened because the machine learning

# classifier requires that each sample is a 1D array

# therefore the 20\*20 image becomes 1\*400

# in machine learning terms that's 400 features with each pixel

# representing a feature

flat\_bin\_image = binary\_image.reshape(-1)

image\_data.append(flat\_bin\_image)

target\_data.append(each\_letter)

return (np.array(image\_data), np.array(target\_data))

def cross\_validation(model, num\_of\_fold, train\_data, train\_label):

# this uses the concept of cross validation to measure the accuracy

# of a model, the num\_of\_fold determines the type of validation

# e.g if num\_of\_fold is 4, then we are performing a 4-fold cross validation

# it will divide the dataset into 4 and use 1/4 of it for testing

# and the remaining 3/4 for the training

accuracy\_result = cross\_val\_score(model, train\_data, train\_label,

cv=num\_of\_fold)

print("Cross Validation Result for ", str(num\_of\_fold), " -fold")

print(accuracy\_result \* 100)

# current\_dir = os.path.dirname(os.path.realpath(\_\_file\_\_))

# training\_dataset\_dir = os.path.join(current\_dir, 'train')

print('reading data')

training\_dataset\_dir = './train20X20'

image\_data, target\_data = read\_training\_data(training\_dataset\_dir)

print('reading data completed')

# the kernel can be 'linear', 'poly' or 'rbf'

# the probability was set to True so as to show

# how sure the model is of it's prediction

svc\_model = SVC(kernel='linear', probability=True)

cross\_validation(svc\_model, 4, image\_data, target\_data)

print('training model')

# let's train the model with all the input data

svc\_model.fit(image\_data, target\_data)

import pickle

print("model trained.saving model..")

filename = './finalized\_model.sav'

pickle.dump(svc\_model, open(filename, 'wb'))

print("model saved")

* + 1. CHARACTER RECOGNITION

This process involves the use of a database of characters or templates. There exists a template for all possible input characters. Templates are created for each of the alphanumeric characters (from A-Z and 0-9). Mean squared error method is used for image similarity. Each segmented character of the license plate is compared with all the standard template characters and the error is recorded. The template corresponding to the least error value is considered as the matched character. Finally the output is displayed on the monitor.

#Code\_for\_PREDICTCHARACTERS

import SegmentCharacters

import pickle

print("Loading model")

filename = './finalized\_model.sav'

model = pickle.load(open(filename, 'rb'))

print('Model loaded. Predicting characters of number plate')

classification\_result = []

for each\_character in SegmentCharacters.characters:

# converts it to a 1D array

each\_character = each\_character.reshape(1, -1);

result = model.predict(each\_character)

classification\_result.append(result)

print('Classification result')

print(classification\_result)

plate\_string = ''

for eachPredict in classification\_result:

plate\_string += eachPredict[0]

print('Predicted license plate')

print(plate\_string)

# it's possible the characters are wrongly arranged

# since that's a possibility, the column\_list will be

# used to sort the letters in the right order

column\_list\_copy = SegmentCharacters.column\_list[:]

SegmentCharacters.column\_list.sort()

rightplate\_string = ''

for each in SegmentCharacters.column\_list:

rightplate\_string += plate\_string[column\_list\_copy.index(each)]

print('License plate')

print(rightplate\_string)

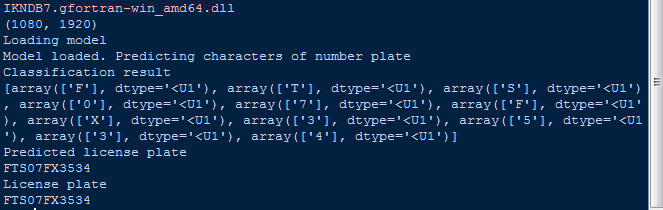


Fig 6.4.4 Character Recognition

# CHAPTER 7 TESTING AND INTEGRATION

## TEST CASE DESCRIPTION



Fig 7.1 Input Image

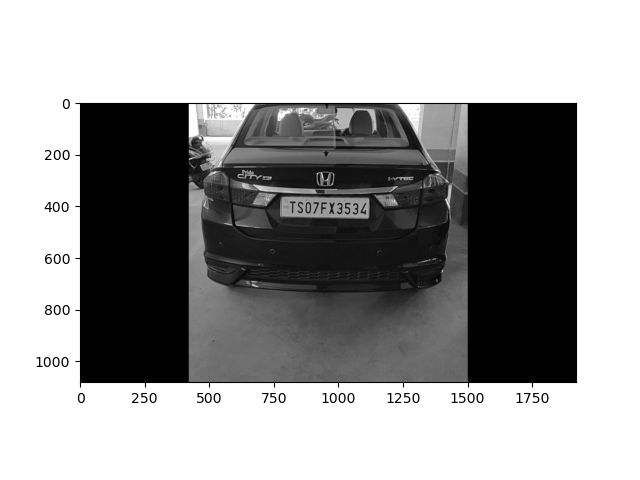


Fig 7.2 Gray scale image

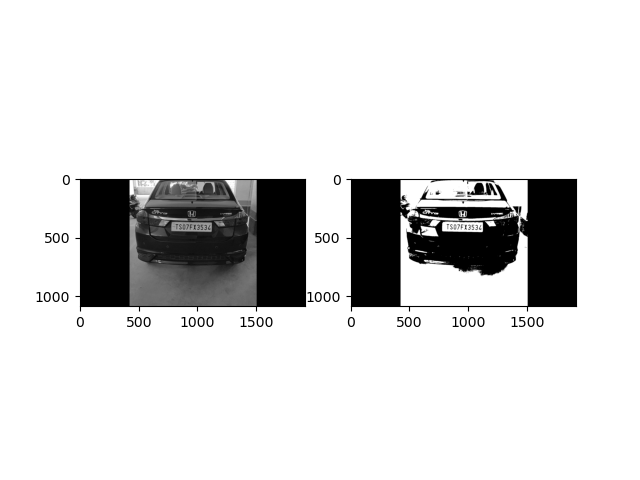


Fig 7.3 Sobel Image

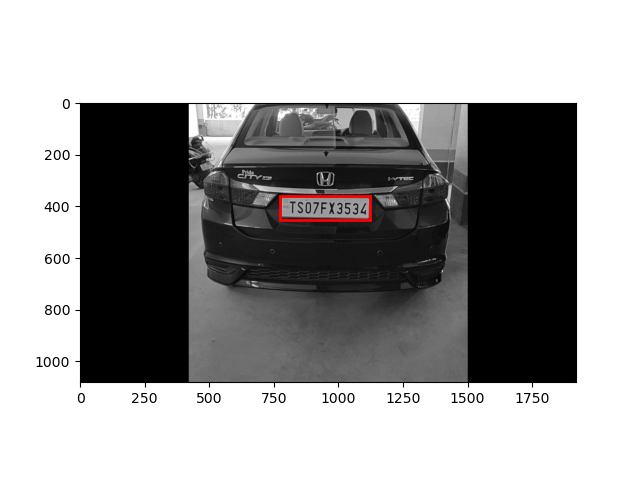


Fig 7.4 Extraction of number plate

:

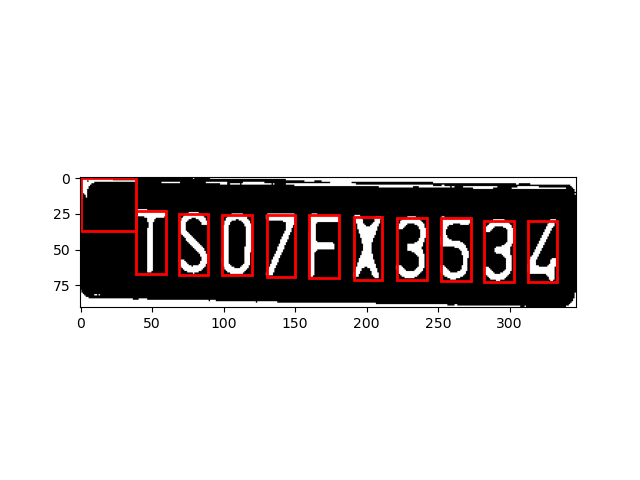
.

Fig 7.5 Segmentation of characters

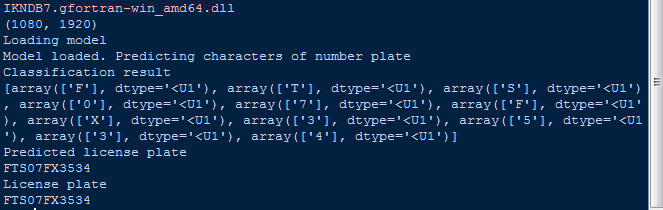


Fig 7.6 Final Output(Prediction of number plate)

Successful tested.

## TYPES OF TESTING

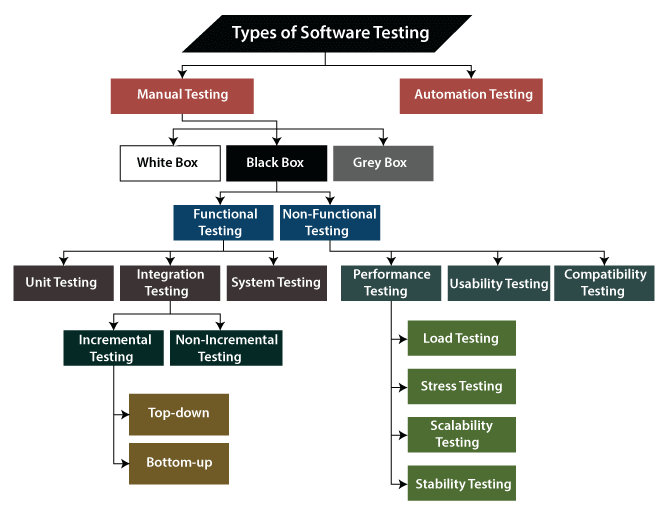


Fig 6.2.1 – Types of Software Testing

## FUTURE ENHANCEMENTS

ANPR can be further exploited for vehicle owner identification, vehicle model identification traffic control, vehicle speed control and vehicle location tracking. It can be further extended as multilingual ANPR to identify the language of characters automatically based on the training data It can provide various benefits like traffic safety enforcement, security- in case of suspicious activity by vehicle, easy to use, immediate information availability- as compare to searching vehicle owner registration details manually and cost effective for any country .

# conclusion

ALPR applications are becoming increasingly complex in Indian context with the phenomenal exponential growth in car, two-wheeler and auto Industries. ALPR applications like automatic toll collection, automatic charging system in parking spaces, management vehicles in parking spaces, and traffic monitoring, etc., have posed new research tasks in ALPR with newer dimensions. We have developed a software for automatic license plate recognition by taking inputs from live video feed. Character segmentation has been implemented on extracted number plates. Finally, segmented characters are recognized by using mean squared error method.

However, few frames exceeded the maximum limit but were compensated by the fast processing of other frames. Thus, here kNN was used as a classifier to compute the classification of the extracted contours. This is done by finding the Euclidean distance between the attributes of the training and validation data set. This data set was nothing but the matrix of pixels from input images fed to the algorithm through the camera. The process starts by segmenting the plate image into individual characters.

These characters are separated by bounding boxes. These boxes were subjected to classification in the kNN algorithm with the help of a template training data set. Moreover, the accuracy of both the system remains the same practically. So, the kNN was at par with accurate pre-existing techniques.

The aim of the paper is to raise the recognition rate of license plate characters through a combination of three main procedures. All trained and tested characters came from the following two main procedures: extracting license plate, segmenting characters. As a rule, the effectiveness of character processing will affect the effect of selected features, and then further affects the efficacy of chosen classifiers. As expected, experimental results show that our proposed combination of three main procedures does give a very high recognition rate, which can be up to 100 % for KNN The system has been tested on static snapshots of vehicles, which has divided into several sets according to difficulties. Sets of 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 100% recognizable set of snapshots, but to test the invariance of the algorithms on random snapshots systematically classified to the sets according to their properties. Currently there are certain restrictions on parameters like speed of the vehicle, script on the number plate, cleanliness of number plate, quality of captured image, skew in the image which can be aptly removed by enhancing the algorithm further.

# REFERENCEs

[1] ‘‘Image Segmentation” .A.D. Jepson and D.J. Fleet, 2007.

[2] Ankush Roy, Debarshi Patanjali Ghoshal, “Number Plate Recognition for Use in Different Countries Using an Improved Segmentation”. 978-1-4244-9581-8/11/$26.00 © 2011 IEEE.

[3] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, “SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving,” in 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), July 2017, pp. 446–454.

[4] C. Gou, K. Wang, Y. Yao, and Z. Li, “Vehicle license plate recognition based on extremal regions and restricted boltzmann machines,” IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1096–1107, April 2016.

[5] G. R. Gonc¸alves, S. P. G. da Silva, D. Menotti, and W. R. Schwartz, “Benchmark for license plate character segmentation,” Journal of Elec- tronic Imaging, vol. 25, no. 5, pp. 053 034–053 034, 2016.

[6] H. Li and C. Shen, “Reading car license plates using deep convolutional neural networks and LSTMs,” CoRR, vol. abs/1601.05610, 2016.

[7] H. Li, P. Wang, and C. Shen, “Towards end-to-end car license plates detection and recognition with deep neural networks,” CoRR, vol. abs/1709.08828, 2017.

[8] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, “ImageNet: A largescale hierarchical image database,” in 2009 IEEE Conference on Computer Vision and Pattern Recognition, June 2009, pp. 248– 255.

[9] Loumos, and E. Kayafas, “License plate recognition from still images and video sequences: A survey,” IEEE Transactions on Intelligent Transportation Systems, vol. 9, no. 3, pp. 377–391, Sept 2008.

[10] O. Bulan, V. Kozitsky, P. Ramesh, and M. Shreve, “Segmentation- and annotation-free license plate recognition with deep localization and failure identification,” IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 9, pp. 2351–2363, Sept 2017.

[11] P.ANISHIYA 1, PROF. S. MARY JOANS , „„Number Plate Recognition for Indian Cars Using Morphological”. 2011 International Conference on Information and Network Technology IACSIT Press, Singapore 115 IPCSIT vol.4 (2011) © (2011) IACSIT Press, Singapore.

[12] S. Z. Masood, G. Shu, A. Dehghan, and E. G. Ortiz, “License plate detection and recognition using deeply learned convolutional neural networks,” CoRR, vol. abs/1703.07330, 2017.

[13] “Digital Image Processing”. Written by R.C.Gonjaliz. 2009.

[14] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," Signal Processing Magazine, IEEE, vol. 20, pp. 21-36, 2003.

[15] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, “Automatic license plate recognition (ALPR): A state-of-the-art review,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 2, pp. 311– 325, Feb 2013.

[16] S.Kranthi, K.Pranathi, A.Srisaila, „„Automatic Number Plate Recognition”. International Journal of Advancements in Technology http://ijict.org/ ISSN 0976-4860.

[17] T.D. Duan, T.L.H.Du, T.V. Phuoc, N.V. Hoang “Building an Automatic Vehicle License-Plate Recognition System” International Conference in Computer Science, Feb 2005.

[18] Wong Eng Yong, „„Vehicle License Plate Registration Recognition System”. Jul07.