A Project Report

On

Automatic Number Plate Detection Using Deep Learning

Submitted for partial fulfillment of requirement for the degree of

Bachelor of Engineering

(Computer Science & Engineering)

Submitted By:

Mr. Prathamesh Sawaikar *Mr. Akshay Dhore*

Mr. Shubham Hushe *Mr. Vaibhav Khandar*

Under the Guidance of

Prof. V. R. Tripathi

Department of Computer Science &Engineering

College of Engineering & Technology,

Akola.

2021-22

Certificate

This is to certify that the project entitled

"Automatic Number Plate Detection Using Deep Learning"

is a bonafide work and it is submitted to the Sant Gadge Baba Amravati University, Amravati

By:

Mr. Shubham Hushe | *M. Akshay Dhore*

Mr. Prathamesh Sawaikar | *Mr. Vaibhav Khandar*

in the partial fulfillment of the requirement for the degree of ***Bachelor of Engineering*** *in* ***Computer Science & Engineering****, during the academic year 2021-22 under my guidance.*

Prof. V. R. Tripathi

Guide,
Dept. of C.S.E.

.

Dr. S. L. Satarkar

Head,
Dept. of C.S.E.

Department of Computer Science &Engineering

College of Engineering & Technology,

Akola.

ACKNOWLEDGEMENT

It is a matter of great pleasure to highlight a fraction of knowledge, we acquired during my technical education through this project. This would not have been possible without the guidance and help of many people. This is where we have the opportunity of expressing gratitude from the core of our heart.

This project would not have been successful without enlightened ideas, timely suggestion and keen interest of my respected Guide ***Prof. V. R. Tripathi***. Being on the same line we express my deep sense of gratitude to ***Dr. S. L. Satarkar***, Head of Department (C.S.E.). We would like to thank ***Dr. S. K. Deshmukh***, Principal of our institution for providing necessary facility during the period of working on this project work.

Thanks are in order to all the colleagues and friends who knowingly or unknowingly helped me during this work.

Mr. Prathamesh Sawaikar
Mr. Akshay Dhore
Mr. Shubham Hushe
Mr. Vaibhav Khandar

ABSTRACT

In this project, a Digital Image Processing-based prototype is developed. Actions such as Image Acquisition, enhancement that is pre-processing, Segmentation of the license plate and then application of OCR (Optical Character Recognition) is applied to store the number on text form. The plate number is displayed as text on the terminal using the principal of OCR with help of pytesseract and Tesseract engine.

It is seen that the security forces and authorities face problems whenever security forces chase a vehicle or they can’t catch a vehicle which broke traffic rules. Authorities find it very hectic on a busy day to log the vehicle numbers manually in a parking lot. So, in order to make the entire process autonomous, we can install this system so as to automatically detect the vehicle which breaks the traffic rules, take a picture of it and store the number in the database so as to fine the respective owner afterwards.

The system can be used in parking so as to take the picture of the vehicle and log the vehicle number in the database (or the cloud, if connected to the internet). This technology reduces the unnecessary hectic manual work if connected to the internet). This technology reduces the unnecessary hectic manual work number of any vehicle once obtained as text, can be displayed, saved in the database or can be searched through the entire database for the details.

This project is so versatile that it can be used as an entire application once converted to a software or can be used as a part of any big project.

INDEX

CHAPTER 1

INTRODUCTION

1.1 IDENTIFICATION OF NEED

In this project, a Digital Image Processing-based prototype is developed. Actions such as Image Acquisition, enhancement that is pre-processing, Segmentation of the license plate and then application of OCR (Optical Character Recognition) is applied to store the number on text form. The plate number is displayed as text on the terminal using the principal of OCR with help of pytesseract and Tesseract engine.

It is seen that the security forces and authorities face problems whenever security forces chase a vehicle or they can’t catch a vehicle which broke traffic rules. Authorities find it very hectic on a busy day to log the vehicle numbers manually in a parking lot.

So, in order to make the entire process autonomous, we can install this system so as to automatically detect the vehicle which breaks the traffic rules, take a picture of it and store the number in the database so as to fine the respective owner afterwards. The system can be used in parking so as to take the picture of the vehicle and log the vehicle number in the database (or the cloud, if connected to the internet).

This technology reduces the unnecessary hectic manual work required on any busy day, saves the labour cost and is far more efficient than humans. The number of any vehicle once obtained as text, can be displayed, saved in the database or can be searched through the entire database for the details.

This project is so versatile that it can be used as an entire application once converted to software or can be used as a part of any big project.

1.2 BACKGROUND

Automatic license plate recognition (ALPR) plays an important role in numerous applications such as unattended parking lots security control of restricted areas traffic law enforcement congestion pricing and automatic toll collection. Due to different working environments, LPR techniques vary from application to application.

Most previous works have in some way restricted their working conditions, such as limiting them to indoor scenes, stationary backgrounds fixed illumination, prescribed driveways limited vehicle speeds or designated ranges of the distance between camera and vehicle.

The aim of this study is to lessen many of these restrictions. Of the various working conditions, outdoor scenes and non-stationary backgrounds may be the two factors that most influence the quality of scene images acquired and in turn the complexity of the techniques needed. In an outdoor environment, illumination not only changes slowly as daytime progresses, but may change rapidly due to changing weather conditions and passing objects (e.g., cars, airplanes, clouds, and overpasses).

In addition, pointable cameras create dynamic scenes when they move, pan or zoom. A dynamic scene image may contain multiple license plates or no license plate at all. Moreover, when they do appear in an image, license plates may have arbitrary sizes, orientations and positions. And, if complex backgrounds are involved, detecting license plates can become quite a challenge.

ANPR systems generally comprises of a camera, software to compare the transformed license plate characters to databases in the system and a user interface to display the images captured with results of transformation. A license plate recognition system generally works in four main parts namely image acquisition, license plate detection, character segmentation and character recognition.

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.

In India, vehicle number plates do not follow a standard language, font or size. Due to the variations in representation of number plates, vehicle number plate extraction, segmentation and recognition are crucial. This demonstration considers vehicle number plates which can contain English characters and numbers only. The system works satisfactorily for wide variation of condition and different type of vehicle plates. The system is implemented and executed in Python OpenCV and performance is tested on genuine images.

1.3 MOTIVATION

The main purpose of this project is to detect a license plate from a video provided by a camera. An efficient algorithm is developed to detect a license plate in various luminance conditions. This algorithm extracts the license plate data from an image and provides it as an input to the stage of Car License Plate Recognition. Extracted image of the number plate can be seen on monitor. 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

1.4 PRELIMINARY INVESTIGATION

Automatic Number Plate Recognition systems are a proven solution for various security forces and administrative authorities around the world. The automatic number plate recognition (ANPR) system market in 2016 was valued at USD 1.78 Billion and is expected to reach USD 3.57 Billion by 2023, at a CAGR (Compound Annual Growth Rate) of 9.74% between 2017 and 2023. The base year considered for the study is 2016 and the forecast period is between 2017 and 2023.

The research methodology used to estimate and forecast the ANPR system market begins with capturing data on key vendor revenue through secondary research. The vendor offerings are considered to determine the market segmentation. This report provides a detailed analysis of the ANPR system market based on type, component, application, and geography.

Market breakdown and data triangulation procedures have been employed to complete the overall market engineering process and arrive at the exact statistics for all segments and sub segments. The breakdown of profiles of primaries is depicted in the figure below:

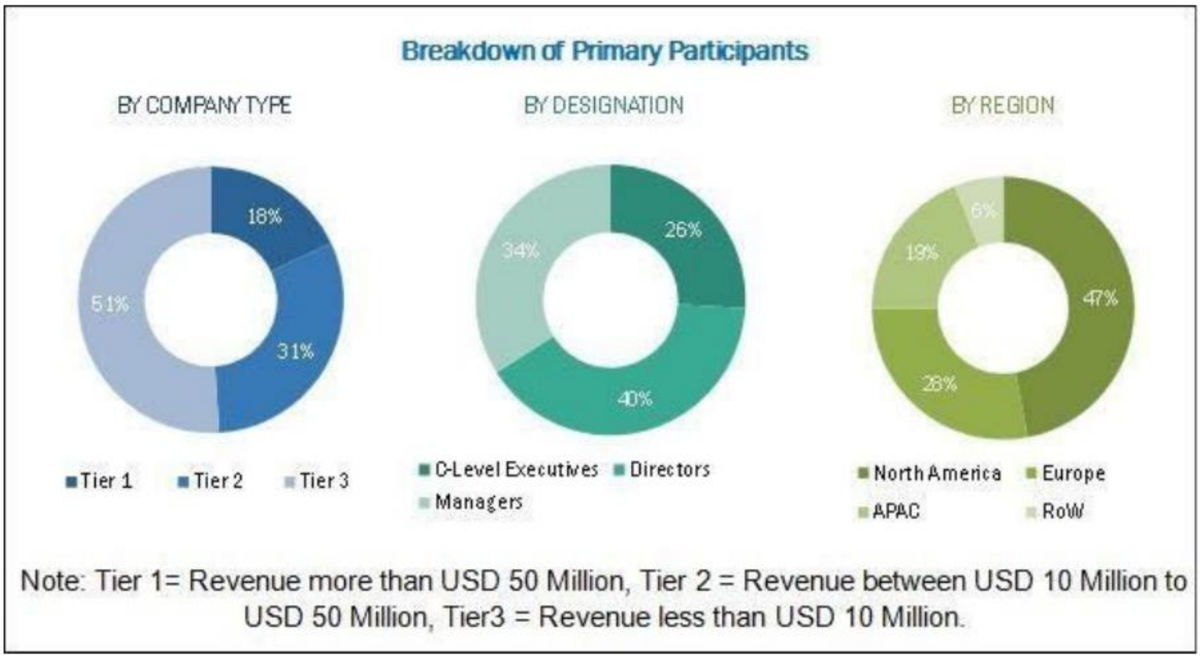


FIGURE 1. BREAKDOWN OF PRIMARY PARTICIPANTS

The traffic management application accounted for the largest market share in 2016. The increasing demand for ANPR systems in urban areas due to high traffic congestion is driving the growth of the ANPR system market for traffic management applications. The market for the electronic toll collection application is expected to grow at the highest rate between 2017 and 2023.

The increasing adoption of vehicles and stringent government regulations by various countries for implementing electronic toll collection systems is driving the growth of the ANPR system market.

After arriving at the overall market size, the total market has been split into several segments and sub segments, which have been verified through the primary research by conducting extensive interviews of people holding key positions such as CEOs (Chief Executive Officers), VPs (Vice Presidents), directors, and executives.

CHAPTER 2

LITERATURE SURVEY

2.1 PREVIOUS WORK

Many developments in Digital Image Processing have been utilized in various fields with advances in Optical Character Recognition Technology as well. Various techniques of employing digital image processing have been developed in recent years. In the 2000s, OCR was made available online as a service (WebOCR), in a cloud computing environment, and in mobile applications like real-time translation of foreign-language signs on a smartphone.

The best application of this technology would be to create a reading machine for the blind, which would allow blind people to have a computer read text to them out loud.

Various commercial and open source OCR systems are available for most common writing systems, including Latin, Cyrillic, Arabic, Hebrew, Indic, Bengali (Bangla), and Devanagari, Tamil, Chinese, Japanese, and Korean characters. The OCR engine used here is Tesseract OCR. Tesseract is an optical character recognition engine for various operating systems. It is free software, released under the Apache License.

Originally developed by Hewlett-Packard as proprietary software in the 1980s, it was released as open source in 2005 and development has been sponsored by Google since 2006. In 2006, Tesseract was considered one of the most accurate open-source OCR engines then available. The Tesseract engine was originally developed as proprietary software at Hewlett Packard labs in Bristol, England and Greeley, Colorado between 1985 and 1994, with some more changes made in 1996 to port to Windows, and some migration from C to C++ in 1998. A lot of the code was written in C, and then some more was written in C++. Since then all the code has been converted to at least compile with a C++ compiler.

Very little work was done in the following decade. It was then released as open source in 2005 by Hewlett Packard and the University of Nevada, Las Vegas (UNLV). Tesseract development has been sponsored by Google since 2006. Tesseract was in the top three OCR engines in terms of character accuracy in 1995. It is available for Linux, Windows and Mac OS X. However, due to limited resources it is only rigorously tested by developers under Windows and Ubuntu.

2.2 EXISTING SYSTEM

Few Inconveniences of the Current Framework are

1) Constant human mediation.

2) High cost.

3) More Manpower is required.

2.3 PROPOSED SYSTEM

The proposed system overcomes the above Disadvantages and apart from them has the beneath specified benefits:

1) Automated framework requiring less labour.

2) Number is displayed and with some modification can be stored in a database or be searched or processed.

3) The featured number plate is automatically cropped and displayed separately.

2.4 HISTORY

Digital Image Processing means processing digital image by means of a digital computer. We can also say that it is a use of computer algorithms, in order to get enhanced image either to extract some useful information. One of the first applications of digital image was in the newspaper industry, when pictures were first sent by submarine cable between London and New York.

Introduction of the Bart lane cable picture transmission system in the early 1920s reduced the time required to transport a picture across the Atlantic from more than a week to less than three hours. Specialized printing equipment coded pictures for cable transmission and then reconstructed them at the receiving end. Some of the initial problems in improving the visual quality of these early digital pictures were related to the selection of printing producers and the distribution of intensity levels.

In fact, digital images require so much storage and computation power that progress in the field of digital image processing has been dependent on the development of digital computers and of supporting technologies that include data storage, display and transmission. The digital image is composed of a finite number of elements, each of which has a location and values. These elements are referred to as picture elements, image elements, pixels or pels.

Pixels used to denote the element of a digital image. The process of acquiring an image of the area containing the text, pre-processing that image, extracting the individual characters, describing the character in the form suitable for computer processing & recognizing those individual characters are Digital Image Processing.

Digital image processing techniques began in the late 1960s and early 1970s to be used in medical imaging, remote Earth resource observations and astronomy. The invention in the early 1970s of computerized axial tomography (CAT) also called computerized tomography (CT) is one of the most important events of image processing in medical diagnosis. Computerized axial tomography is a process in which a ring of detectors encircles a patient and an X-Ray source, concentric with a detector ring, rotates about the patient. The X-ray passes through the object and is collected at the opposite end by the corresponding detectors in the ring.

As the source rotates, this procedure is repeated. Image processing methods have successfully restored blurred pictures that were the only available records of rare artefacts lost or damaged after being photographed. Image processing methods have successfully restored blurred pictures that were the only available records of rare artefacts lost or damaged after being photographed.

Early optical character recognition may be traced to technologies involving telegraphy and creating reading devices for the blind. In 1914, Emanuel Goldberg developed a machine that read characters and converted them into standard telegraph code. Concurrently, Edmund Fournier d'Albe developed the Optophone, a handheld scanner that when moved across a printed page, produced tones that corresponded to specific letters or characters. In the late 1920s and into the 1930s Emanuel Goldberg developed what he called a "Statistical Machine" for searching microfilm archives using an optical code recognition system.

In 1931 he was granted USA Patent number 1,838,389 for the invention. The patent was acquired by IBM. With the advent of smart-phones and smart glasses, OCR can be used in internet connected mobile device applications that extract text captured using the device's camera. These devices that do not have OCR functionality built into the operating system will typically use an OCR API to extract the text from the image file captured and provided by the device.

The OCR API returns the extracted text, along with information about the location of the detected text in the original image back to the device app for further processing (such as text-to-speech) or display. In 1951, a young Department of Defence engineer named David Shepard developed a scanning device in his home that he nicknamed ‘Gismo.’ This device could read twenty-three letters of the alphabet, interpret Morse Code, and read aloud letter by letter.

While crude by today’s standards, it nevertheless captured the imagination of scientists and businessmen alike with its potential as a practical business tool in the data entry field. ‘Gismo’ garnered quite a bit of publicity and consequently spurred on the further development of OCR tools.

CHAPTER 3

PROPOSED METHODOLOGY

3.1 PROBLEM FORMULATION

It is seen that the security forces and authorities face problems whenever security forces chase a vehicle or they can't catch a vehicle which broke traffic rules. Authorities find it very hectic on a busy day to log the vehicle numbers manually in a parking lot. So, in order to make the entire process autonomous, we can install this system so as to automatically detect the vehicle which breaks the traffic rules, take a picture of it and store the number in the database so as to fine the respective owner afterwards. The system can be used in parking so as to take the picture of the vehicle and log the vehicle number in the database (or the cloud, if connected to the internet). This technology reduces the unnecessary hectic manual work required on any busy day, saves the labour cost and is far more efficient than humans. The number of any vehicle once obtained as text, can be displayed, saved in the database or can be searched through the entire database for the details. This project is so versatile that it can be used as an entire application once converted to a software or can be used as a part of any big project.

3.2 PROJECT OBJECTIVE

1) Image Acquisition using the computer's primary camera.

2) Image Enhancement and pre-processing to improve the quality of the image and convert the image to binary scale so as to use it in contour extraction.

3) Extract the number plate region from the binary image and display it separately.

4) Apply optical character recognition to display the license plate number from the picture as text.

In the input unit, a preconfigured camera is arranged such that it is interfaced with a display device like a TFT monitor and a switch or a key is attached. Whenever the subject (the car) is in the frame of the picture, the key is pressed in order to capture the picture.

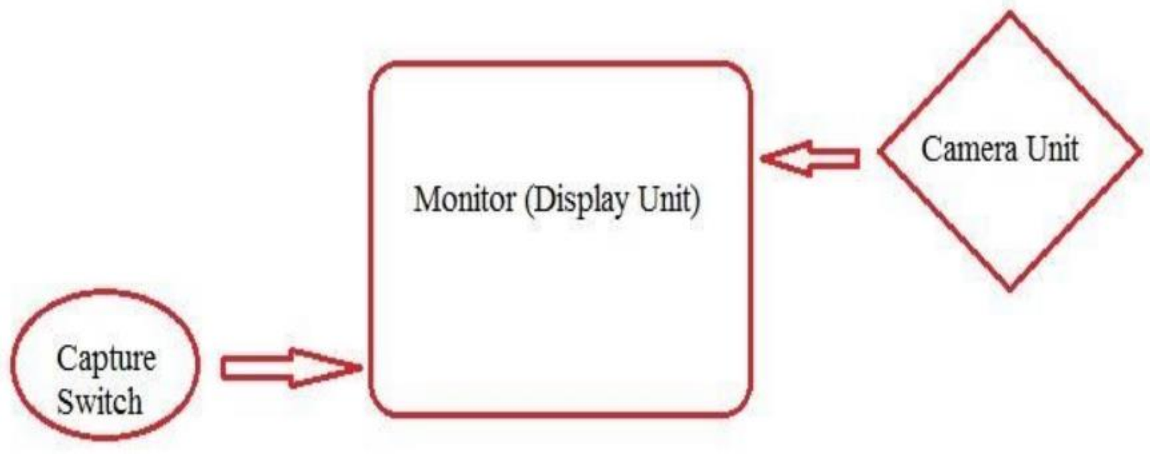


FIGURE 2. BLOCK DIAGRAM OF INPUT UNIT

In the Output Unit, the input image is taken from the input unit and then processed in then the number on the license plate is extracted using Optical Character Recognition and then it can either be stored in a database or be displayed on a display device or both or can be used to excite an actuator.

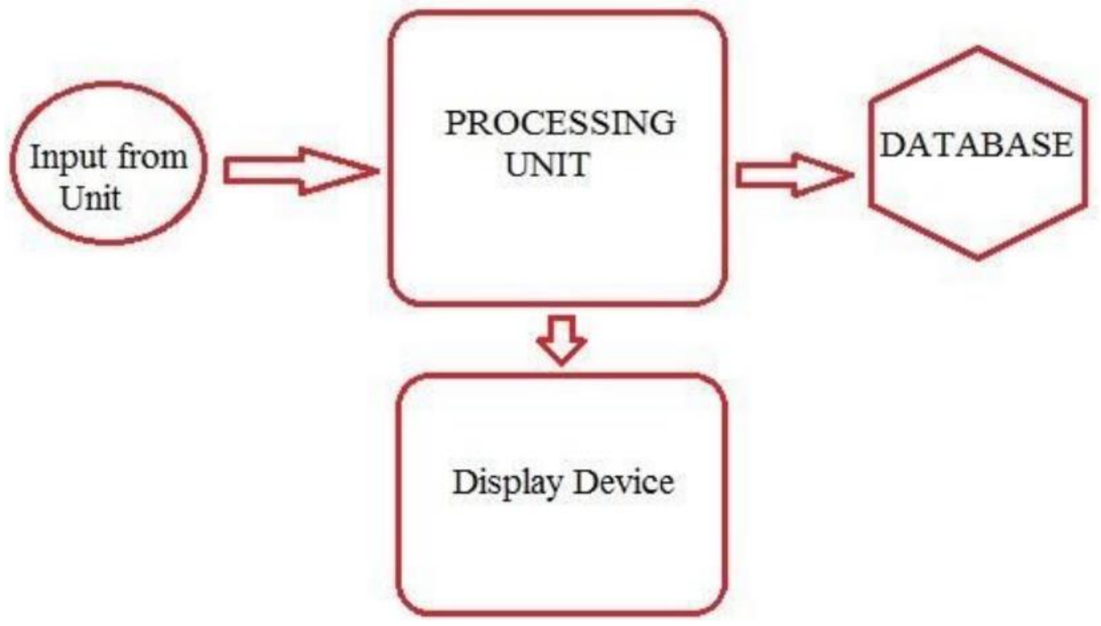


FIGURE 3. BLOCK DIAGRAM OF OUTPUT UNIT

3.3 DETAILS OF PROCESSING

Basics of Digital Image Processing : The image of a vehicle whose number plate is to be recognised is taken from a digital camera which is then loaded to a local computer for further processing. Open CV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. In simple language it is a library used for Image Processing. It is mainly used to do all the operations related to Images. Python, being a versatile language, is used here as a programming language. Python and its modules like Numpy, Scipy, Matplotlib and other special modules provide the optimal functionality to be able to cope with the flood of pictures. To enhance the number plate recognition further, we use a median filter to eliminate noises but it not only eliminates noise. It concentrates on high frequency also. So it is more important in edge detection in an image, generally rectangular plate. Image Processing mainly involves the following steps:

1. Image acquisition: This is the first step or process of the fundamental steps of digital image processing. Image Acquisition is the capturing of an image by any physical device (in this case the primary camera of the computer) so as to take the input as a digital image in the computer.
2. Image Enhancement: Image enhancement is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement techniques is to bring out detail that is obscured, or simply to highlight certain features of interest in an image. Such as, changing brightness & contrast etc. In this step the quality or rather the clarity of the input image is enhanced and the image is made clear enough to be processed.
3. Morphological Processing: Morphological operations apply a structuring element to an input image, creating an output image of the same size. The image is converted to a

binary image, making it more to apply structural extraction to the image and extract any structure related to a particular mathematical model from it, in this case a license plate.

4. Segmentation: Segmentation procedures partition an image into its constituent parts or objects. In general, autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged segmentation procedure brings the process a long way toward a successful solution of imaging problems that require objects to be identified individually.
5. Representation: Representation and description almost always follow the output of a segmentation stage, which usually is raw pixel data, constituting either the boundary of a region or all the points in the region itself. Choosing a representation is only part of the solution for transforming raw data into a form suitable for subsequent computer processing. Description deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another.
6. Recognition: Recognition is the process that assigns a label, such as, “Plate” to an object based on its descriptors.

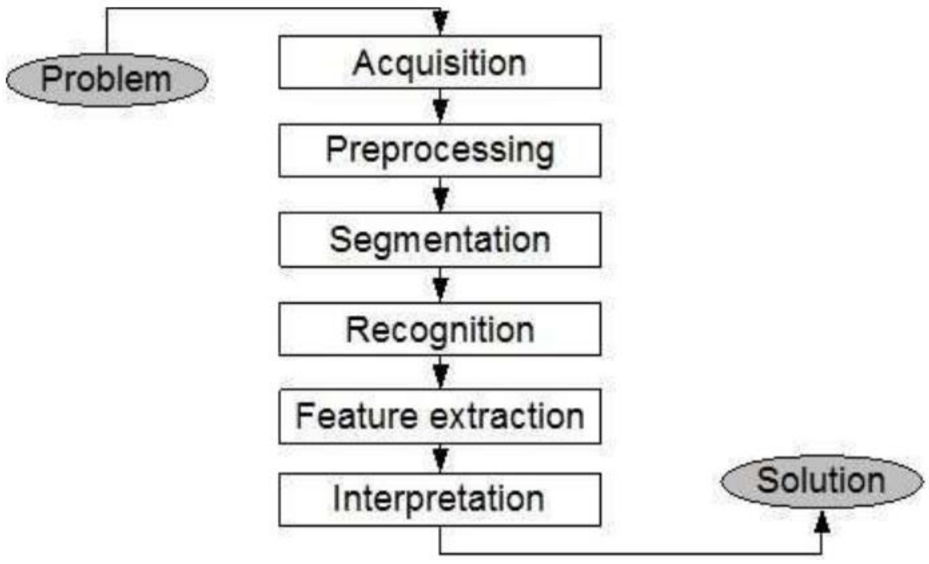


FIGURE 4. DETAILS OF PROCESSING

Optical Character Recognition: Optical character recognition or optical character reader, often abbreviated as OCR, is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast). The Techniques in OCR involve:

1) Pre-Processing
2) Character Recognition
3) Post-Processing

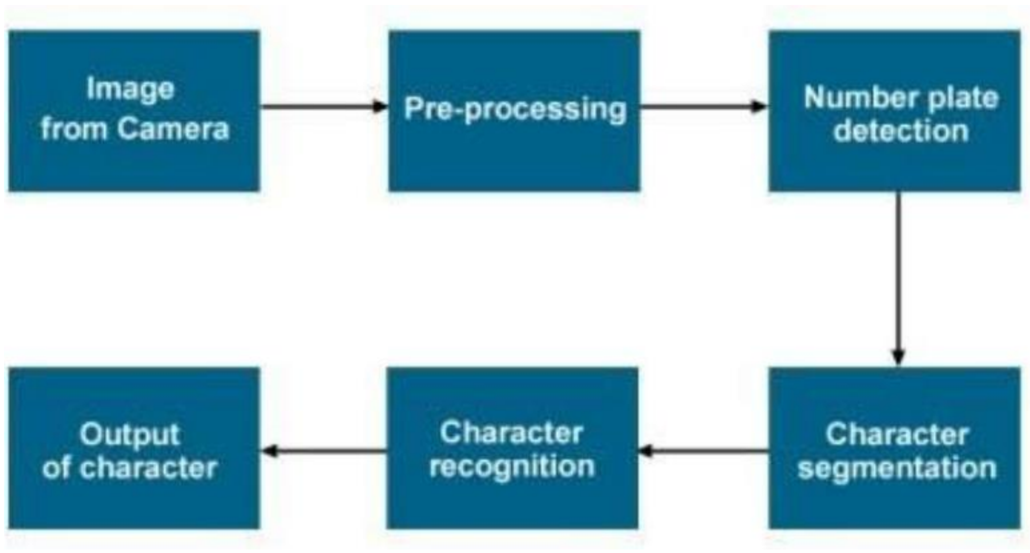


FIGURE 5. TECHNIQUES IN OCR

3.4 WORKING OF THE PROPOSED METHODOLOGY

- In this project, a prototype of Digital Image processing and OCR are executed using different python libraries such as Open CV 4.2 and py-tesseract respectively.
- The primary camera of the computer is accessed and the image is clicked after any key is pressed, when the vehicle is in the frame.
- The input image is then fed in the system, for further processing.
- The Morphed image and the input image and the morphed image is then displayed when any key is pressed. The Morphed image is obtained after morphological transformation.
- After pressing one more key, the segmented plate is displayed from the morphed image in a new window which is performed using contour extraction on the morphed image.

- The final step involves performing optical character recognition on the segmented plate using Tesseract engine and a library known as py-tesseract in python. The vehicle number is displayed on the terminal and the plate region is highlighted in new image in a new window, after pressing one more key.

3.5 PREREQUISITES OF THE SYSTEM

3.5.1 COMPUTER SPECIFICATION:

- Disk space: 1 GB
- Operating systems: Windows 10 or later, MacOS, and Linux
- Python versions: 3.9.6 and above
- Included development tools: conda, conda-env, Jupyter Notebook (IPython)
- Compatible tools: Microsoft Visual Studio, PyCharm
- Included Python packages: NumPy, SciPy, scikit-learn, pandas, Matplotlib, Numba, Intel Threading Building Blocks, pyDAAL, Jupyter, mpi4py, PIP, and others.
- Processors: Intel Core i3 processor or later
- Camera set as primary camera

3.5.2 REQUIRED PACKAGES:

Python 3.9.6 and above : Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aims to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organizaxtion, the Python Software Foundation, manages Python and CPython.

Syntax and semantics : Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

Indentation: Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is also sometimes termed the off-side rule.

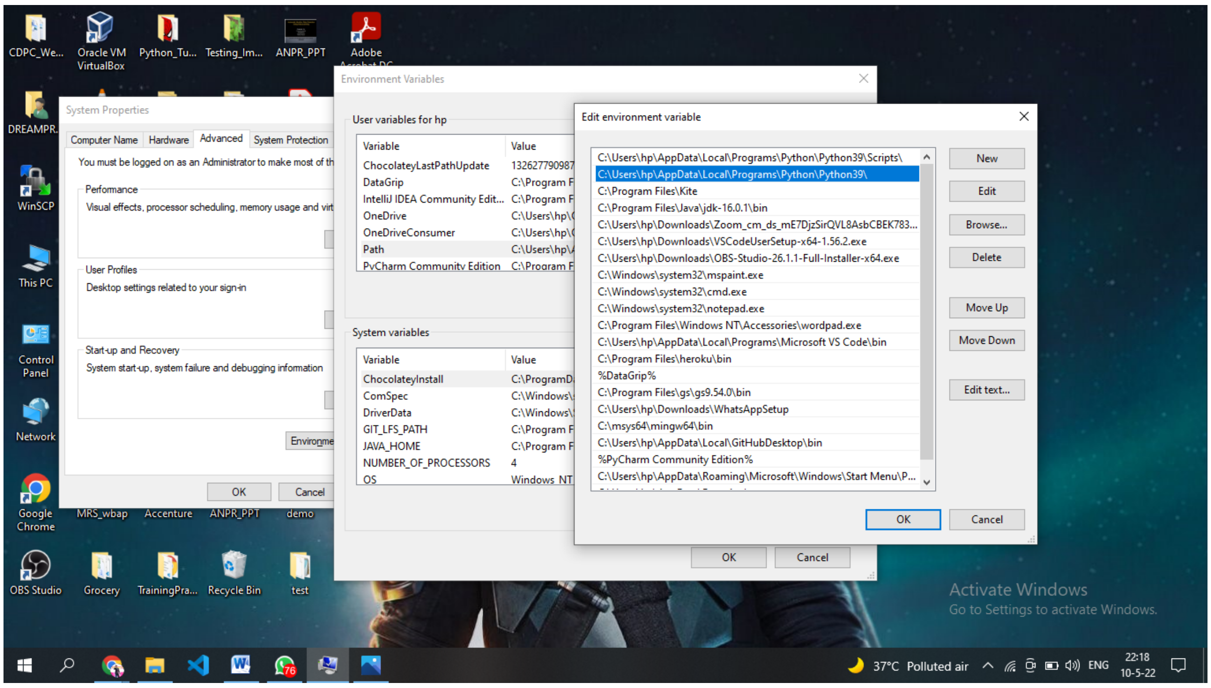
Libraries: Python's large standard library, commonly cited as one of its greatest strengths, provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary precision decimals, manipulating regular expressions, and unit testing.

As of March 2018, the Python Package Index (PyPI), the official repository for third-party Python software, contains over 130,000 packages with a wide range of functionality, including:

1. Graphical user interfaces
2. Web frameworks
3. Multimedia
4. Databases
5. Networking
6. Test frameworks
7. Automation
8. Web scraping
9. Documentation
10. System administration
11. Scientific computing
12. Text processing
13. Image processing

Installation: Python 3.9.6 for Windows 10 64 bit MSI installer can be downloaded from python.org/downloads/release/ and installed accordingly. The same URL contains the installer for various other operating systems but this project is built on Windows 10.

After installation of python, add the path of python to the environment variables of the system by accessing the advanced System settings from the control panel as shown below.



You must be logged on as an Administrator to make most of th

Performance
Visual effects, processor scheduling, memory usage and virt

User Profiles
Desktop settings related to your sign-in

Start-up and Recovery
System start-up, system failure and debugging information

User variables for hp

Variable	Value
ChocolateyLastPathUpdate	132627790987
DataGrip	C:\Program F
IntelliJ IDEA Community Edit...	C:\Program F
OneDrive	C:\Users\hp\(
OneDriveConsumer	C:\Users\hp\(
Path	C:\Users\hp\A
PyCharm Community Edition	C:\Program F

System variables

Variable	Value
ChocolateyInstall	C:\ProgramD
ComSpec	C:\Windows\
DriverData	C:\Windows\
GIT_LFS_PATH	C:\Program F
JAVA_HOME	C:\Program F
NUMBER_OF_PROCESSORS	4
OS	Windows_NT

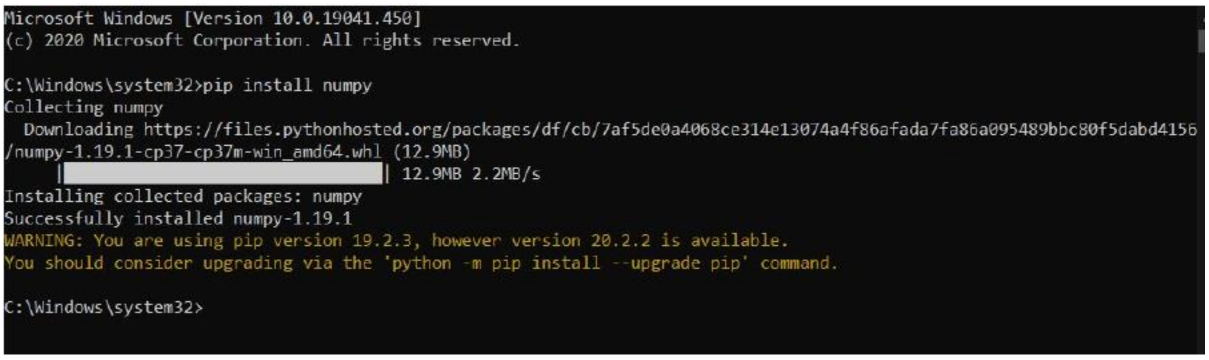
```
C:\Users\hp\AppData\Local\Programs\Python\Python39\Scripts\
C:\Users\hp\AppData\Local\Programs\Python\Python39\
C:\Program Files\Kite
C:\Program Files\Java\jdk-16.0.1\bin
C:\Users\hp\Downloads\Zoom_cm_ds_mE7DjzSirQVL8AsbCBEK783...
C:\Users\hp\Downloads\VSCodeUserSetup-x64-1.56.2.exe
C:\Users\hp\Downloads\OBS-Studio-26.1.1-Full-Installer-x64.exe
C:\Windows\system32\mspaint.exe
C:\Windows\system32\cmd.exe
C:\Windows\system32\notepad.exe
C:\Program Files\Windows NT\Accessories\wordpad.exe
C:\Users\hp\AppData\Local\Programs\Microsoft VS Code\bin
C:\Program Files\heroku\bin
%DataGrip%
C:\Program Files\gs\gs9.54.0\bin
C:\Users\hp\Downloads\WhatsAppSetup
C:\msys64\mingw64\bin
C:\Users\hp\AppData\Local\GitHubDesktop\bin
%PyCharm Community Edition%
C:\Users\hp\AppData\Roaming\Microsoft\Windows\Start Menu\P...
```

Python Install Package (pip): pip is the package installer for Python. You can use pip to install packages from the Python Package Index and other indexes. Installing with get-pip.py To install pip, securely download **get-pip.py: https://bootstrap.pypa.io/get-pip.py** Then run the following: **python get-pip.py** One major advantage of pip is the ease of its command-line interface, which makes installing Python software packages as easy as issuing a command: **pip install some-package-name**

Users can also easily remove the package: **pip uninstall some-package-name**

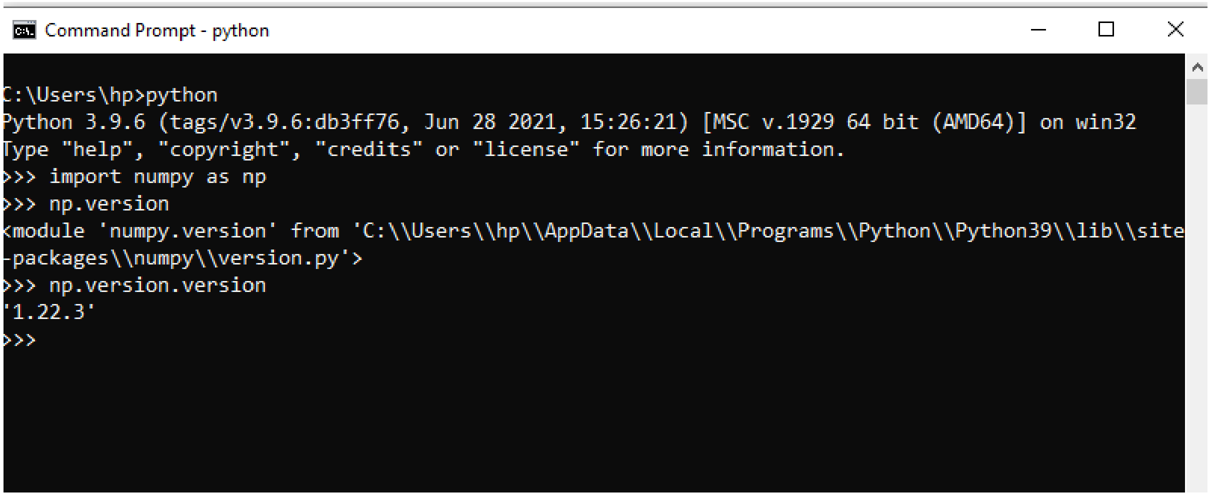
Installation: Numpy 1.19.1 can be installed by using the pip command:

pip install numpy==1.19.1



```
Microsoft Windows [Version 10.0.19041.450]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Windows\system32>pip install numpy
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/df/cb/7af5de0a4068ce314e13074a4f86afada7fa86a095489bbc80f5dabd4156
/numpy-1.19.1-cp37-cp37m-win_amd64.whl (12.9MB)
     |████████████████████████████████| 12.9MB 2.2MB/s
Installing collected packages: numpy
Successfully installed numpy-1.19.1
WARNING: You are using pip version 19.2.3, however version 20.2.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Windows\system32>
```

After successful installation of numpy, one can open the python terminal in the command prompt and import numpy to test the version of numpy installed as follows:



```
C:\Users\hp>python
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> np.version
<module 'numpy.version' from 'C:\\Users\\hp\\AppData\\Local\\Programs\\Python\\Python39\\lib\\site
-packages\\numpy\\version.py'>
>>> np.version.version
'1.22.3'
>>>
```

Numpy: NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

The NumPy library contains multidimensional array and matrix data structures (you'll find more information about this in later sections). It provides **ndarray**, a homogeneous ndimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

Py-tesseract: Humans can understand the contents of an image simply by looking. We perceive the text on the image as text and can read it. Computers don't work the same way. They need something more concrete, organized in a way they can understand. This is where Optical Character Recognition (OCR) kicks in.

Whether it's recognition of car plates from a camera, or hand-written documents that should be converted into a digital copy, this technique is very useful. While it's not always perfect, it's very convenient and makes it a lot easier and faster for some people to do their jobs.

Optical Character Recognition involves the detection of text content on images and translation of the images to encoded text that the computer can easily understand.

An image containing text is scanned and analysed in order to identify the characters in it. Upon identification, the character is converted to machine-encoded text. The image is first scanned and the text and graphics elements are converted into a bitmap, which is essentially a matrix of black and white dots. The image is then pre-processed where the brightness and contrast are adjusted to enhance the accuracy of the process.

The image is now split into zones identifying the areas of interest such as where the images or text are and this helps kickoff the extraction process. The areas containing text can now be broken down further into lines and words and characters and now the software is able to match the characters through comparison and various detection algorithms. '

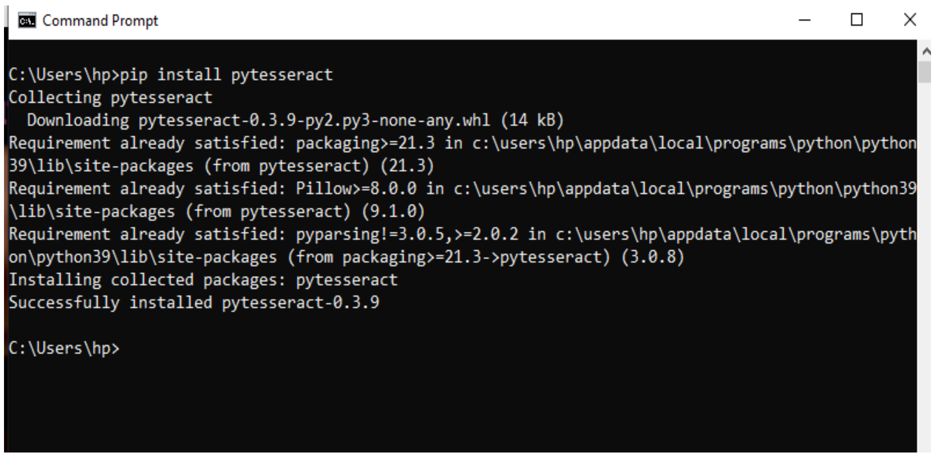
The final result is the text in the image that we're given. The process may not be 100% accurate and might need human intervention to correct some elements that were not scanned correctly. Error correction can also be achieved using a dictionary or even Natural Language Processing (NLP). We will use the Python-Tesseract, or simply Py-Tesseract, library which is a wrapper for Google's Tesseract-OCR Engine.

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images.

Python-tesseract is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Python Imaging Library, including jpeg, png, gif, bmp, tiff, and others, whereas tesseract-ocr by default only supports tiff and bmp.

Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file. Support for OpenCV image/NumPy array objects.

Installing via pip: **pip install pytesseract**



```
C:\Users\hp>pip install pytesseract
Collecting pytesseract
  Downloading pytesseract-0.3.9-py2.py3-none-any.whl (14 kB)
Requirement already satisfied: packaging>=21.3 in c:\users\hp\appdata\local\programs\python\python
39\lib\site-packages (from pytesseract) (21.3)
Requirement already satisfied: Pillow>=8.0.0 in c:\users\hp\appdata\local\programs\python\python39
\lib\site-packages (from pytesseract) (9.1.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\hp\appdata\local\programs\pyth
on\python39\lib\site-packages (from packaging>=21.3->pytesseract) (3.0.8)
Installing collected packages: pytesseract
Successfully installed pytesseract-0.3.9

C:\Users\hp>
```

The whole process of Tesseract-OCR is depicted below:

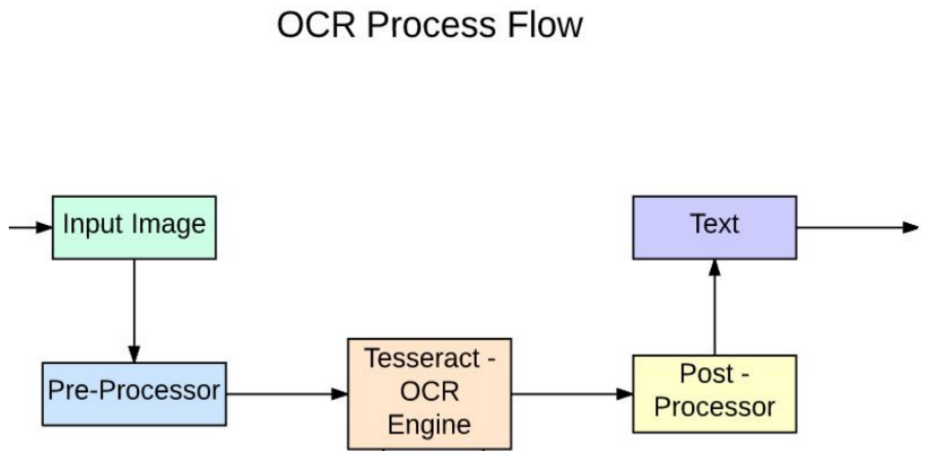


FIGURE 6. OCR PROCESS FLOW

OpenCV: OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

OpenCV has more than 47 thousand people in the user community and an estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV.

OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

OpenCV's application areas include:

1. 2D and 3D feature toolkits
2. Egomotion estimation
3. Facial recognition system
4. Gesture recognition
5. Human–computer interaction (HCI)
6. Mobile robotics
7. Motion understanding
8. Object identification
9. Segmentation and recognition
10. Stereopsis stereo vision: depth perception from 2 cameras
11. Structure from motion (SFM)

OpenCV runs on the following desktop operating systems: Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD. OpenCV runs on the following mobile operating systems: Android, iOS, Maemo, BlackBerry 10. The user can get official releases from SourceForge or take the latest sources from GitHub. OpenCV uses CMake.

Installation: The project uses OpenCV (4.4.0) as the computer vision library and requires some prerequisites as:

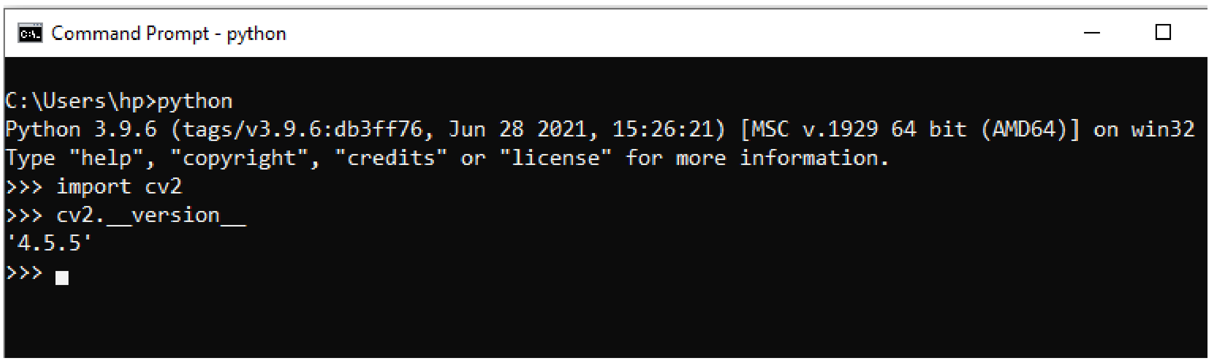
1. Python-3.9.6
2. Numpy
3. Matplotlib (Matplotlib is optional, but recommended since we use it a lot in our tutorials).

Install all packages into their default locations. Python will be installed to C:/Python39/.

After installation, open command prompt and write command:

pip install opencv-python

The version of opencv library installed can be checked as follows:



```
C:\Users\hp>python
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'4.5.5'
>>>
```

CHAPTER 4

DESIGN AND ANALYSIS

4.1 SYSTEM ARCHITECTURE

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.

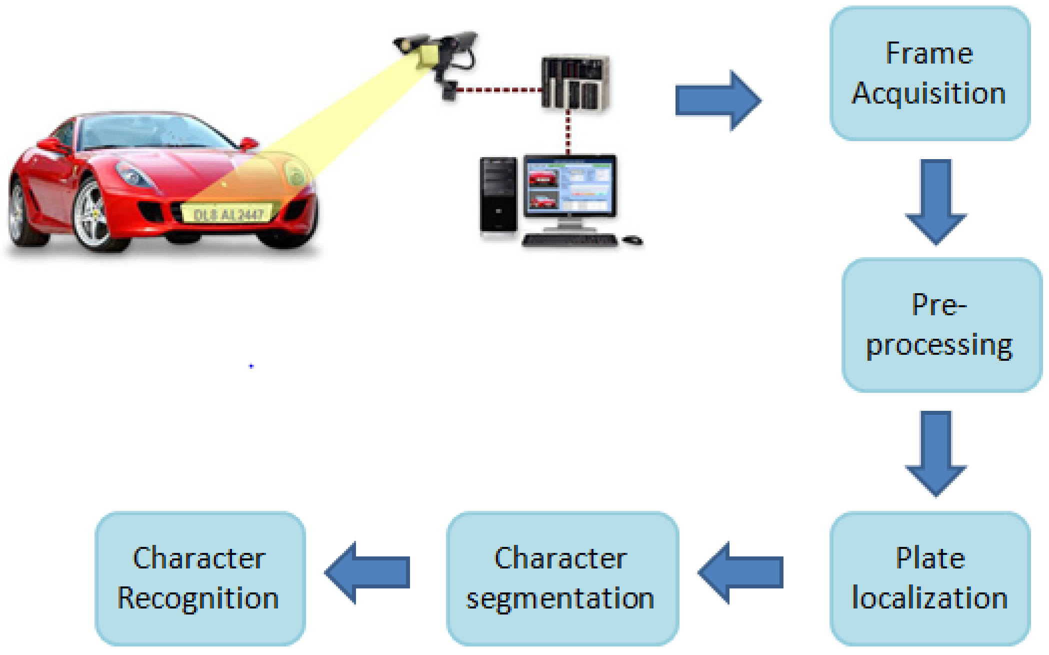


FIGURE 7. SYSTEM ARCHITECTURE

4.2 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.

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.

NORMAL FLOW OF EVENTS:

- The camera captures the video of the vehicle.
- The software identifies and extracts the characters of the number plate of the vehicle and checked for its validity.

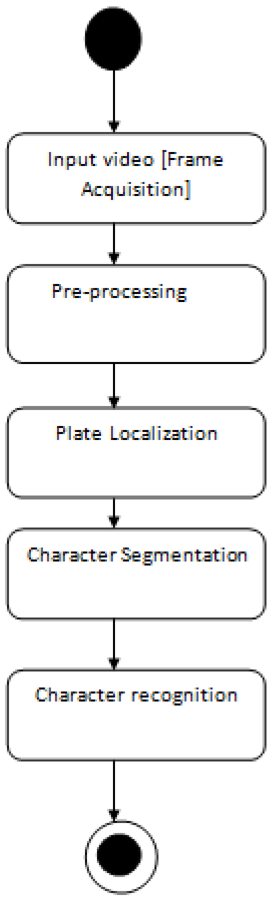


FIGURE 8. ACTIVITY DIAGRAM

4.3 DATAFLOW DIAGRAM

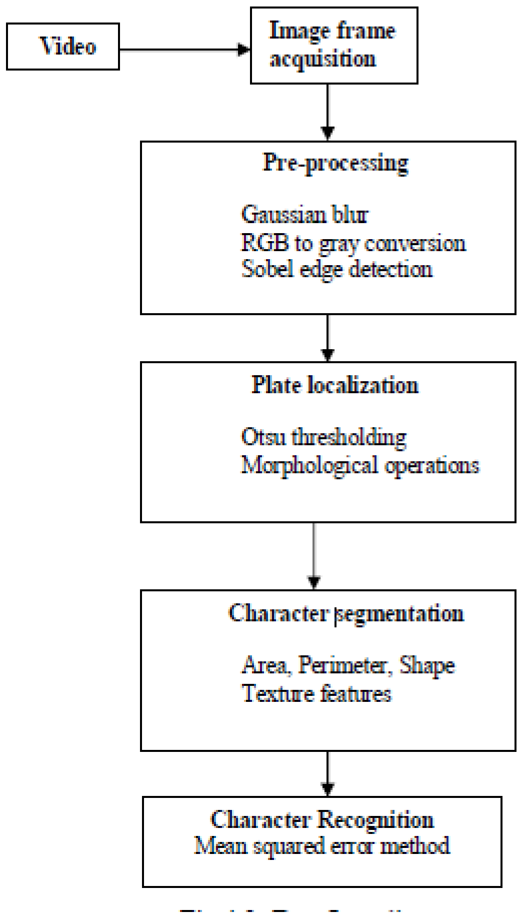


FIGURE 9. DATAFLOW DIAGRAM

4.4 ALGORITHM

Algorithm to Recognize the Number Plates

The sequence of processes associated with Number plate recognition is given below. The file upload process is initiated by the data owner entity.

Input: Uploading the image file from camera

Output: Vehicle number plate in characters

- Read the original image or Capture the image
- Resize the image
- Convert it to grayscale.

- Apply Bilateral Filter. *What is a bilateral filter* ? A bilateral filter is a non-linear, edge preserving, and noise-reducing smoothing filter for images. It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels.
- Identify and store the Canny edges. *What are Canny edges* ? The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images.
- Find the contours in from the edges detected and sort the top 30 contours.
- Get the perimeter of each contour and select those with 4 corners.
- Mask all other parts of the image and show the final image.
- Read the text using Tesseract OCR
- Standardize the text to Indian vehicle number plate format
- Stop

On upload of the image file to the system, the number plate recognition system performs its functions to provide the output.

4.5 CLASS DIAGRAM

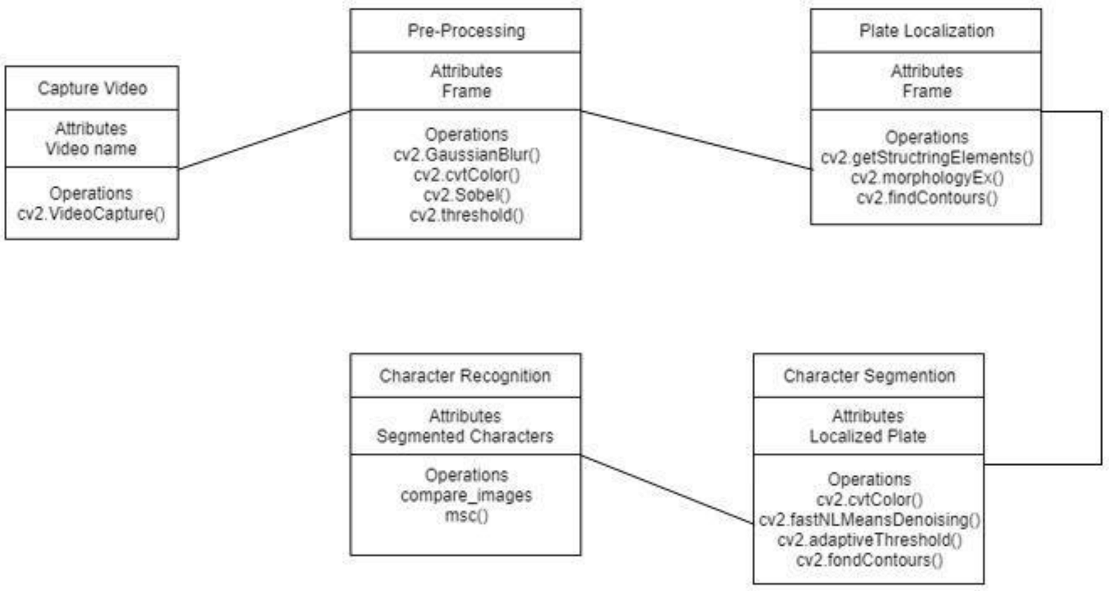


FIGURE 10. CLASS DIAGRAM

CHAPTER 5

OPTICAL CHARACTER RECOGNITION

Optical character recognition or optical character reader (OCR) is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo). Widely used as a form of data entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

FIGURE 12. OPTICAL CHARACTER RECOGNITION

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of recognition accuracy for most fonts are now common, and with support for a variety of digital image file format inputs. Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

5.1 TECHNIQUES OF OCR

Pre-processing

OCR software often "pre-processes" images to improve the chances of successful recognition. Techniques include:

- De-skew – If the document was not aligned properly when scanned, it may need to be tilted a few degrees clockwise or counterclockwise in order to make lines of text perfectly horizontal or vertical.
- Despeckle – remove positive and negative spots, smoothing edges
- Line removal – Cleans up non-glyph boxes and lines
- Layout analysis or "zoning" – Identifies columns, paragraphs, captions, etc. as distinct blocks. Especially important in multi-column layouts and tables.

Text Recognition

Matrix matching involves comparing an image to a stored glyph on a pixel-by-pixel basis; it is also known as "pattern matching", "pattern recognition", or "image correlation". This relies on the input glyph being correctly isolated from the rest of the image, and on the stored glyph being in a similar font and at the same scale. This technique works best with typewritten text and does not work well when new fonts are encountered. This is the technique the early physical photocell-based OCR implemented, rather directly.

Post-processing

OCR accuracy can be increased if the output is constrained by a lexicon – a list of words that are allowed to occur in a document. This might be, for example, all the words in the English language, or a more technical lexicon for a specific field. This technique can be problematic if the document contains words not in the lexicon, like proper nouns. Tesseract uses its dictionary to influence the character segmentation step, for improved accuracy. The output stream may be a plain text stream or file of characters, but more sophisticated OCR systems can preserve the original layout of the page and produce, for example, an annotated PDF that includes both the original image of the page and a searchable textual representation.

5.2 USES OF OCR

- Data entry for business documents, e.g. check, passport, invoice, bank statement and receipt.
- Automatic number plate recognition.
- In airports, for passport recognition and information extraction.
- Automatic insurance documents key information extraction.
- Traffic sign recognition.
- Extracting business card information into a contact list.
- More quickly make textual versions of printed documents, e.g. book scanning for Project Gutenberg.
- Make electronic images of printed documents searchable, e.g. Google Books.
- Converting handwriting in real time to control a computer (pen computing).

- Defeating CAPTCHA anti-bot systems, though these are specifically designed to prevent OCR. The purpose can also be to test the robustness of CAPTCHA anti-bot systems.
- Assistive technology for blind and visually impaired users.
- Writing the instructions for vehicles by identifying CAD images in a database that are appropriate to the vehicle design as it changes in real time.
- Making scanned documents searchable by converting them to searchable PDFs

CHAPTER 6

PROJECT IMPLEMENTATION

- In this project Digital Image processing and OCR are executed using different python libraries such as OpenCV 3 and pytesseract respectively.
- The primary camera of the computer is accessed and the image is clicked after any key is pressed, when the vehicle is in the frame.
- The input image is then fed in the system, for further processing.
- The Morphed image and the input image and the morphed image is then displayed when any key is pressed. The Morphed image is obtained after morphological transformation.
- After pressing one more key, the segmented plate is displayed from the morphed image in a new window which is performed using contour extraction on the morphed image.
- The final step involves performing optical character recognition on the segmented plate using the Tesseract library known as py-tesseract in python. The vehicle number is displayed on the terminal and the plate region is highlighted in a new image in a new window, after pressing one more key.
- The project is tested on various sets of test data and the system is found to work fine with certain pictures with particular frame measurements.

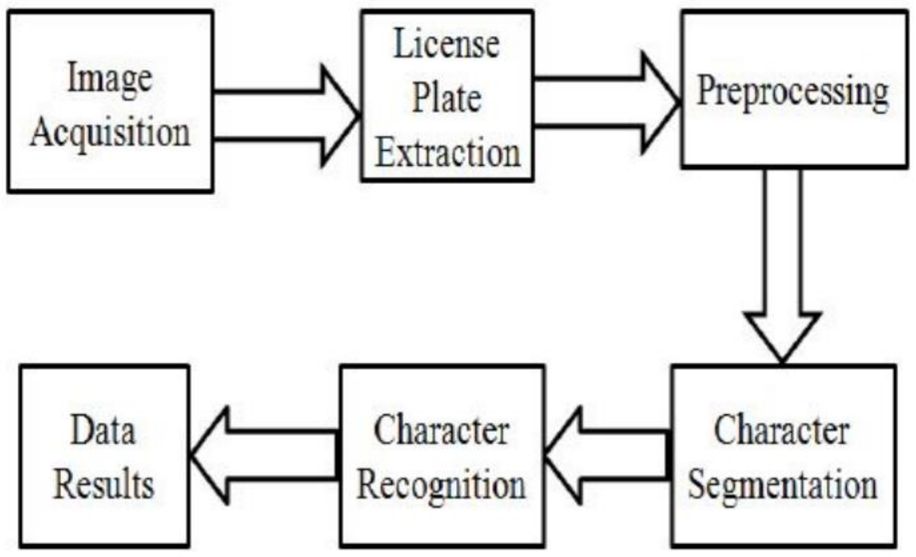


FIGURE 11. PROJECT IMPLEMENTATION

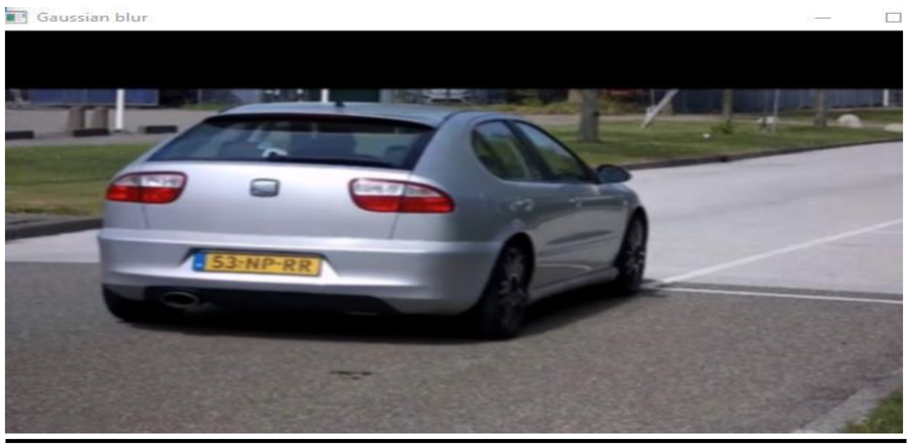
6.1 MODULE DESCRIPTION

The proposed system has the following module:

6.1.1 PREPROCESSING

Pre-processing steps involves the following methods

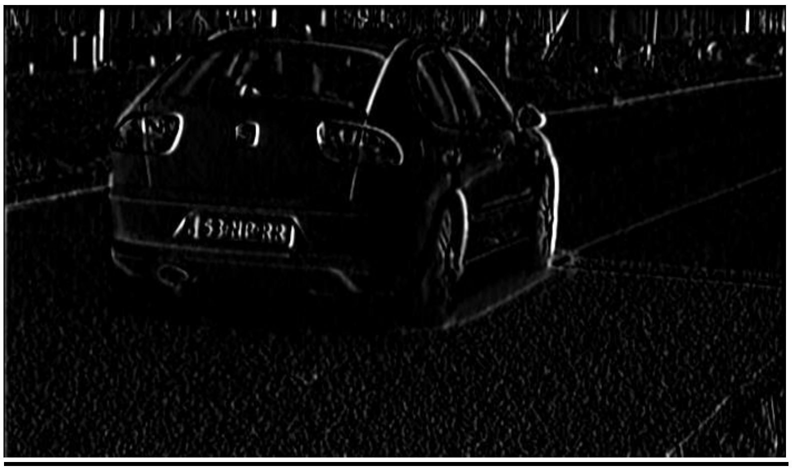
1. Gaussian blur: Gaussian blurring is highly effective in removing gaussian noise from the image. Image blurring is achieved by convolving the image with a low-pass filter kernel. It actually removes high frequency content (e.g: noise, edges) from the image resulting in edges being blurred. In this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



2. Gray scale conversion: It involves conversion of RGB image into grey image.

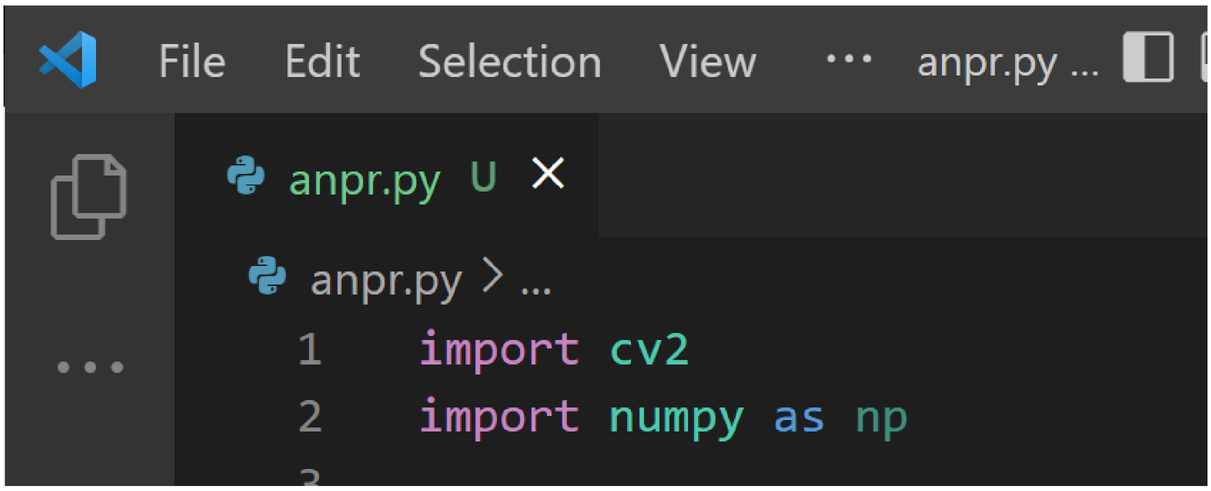
A gray-scale image is composed of different shades of grey color. A true color image can be converted to a gray scale image by preserving the luminance (brightness) of the image. Here the RGB image is a combination of RED, BLUE AND GREEN colors. The grayscale image is obtained from the RGB image by combining 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.

3. 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).



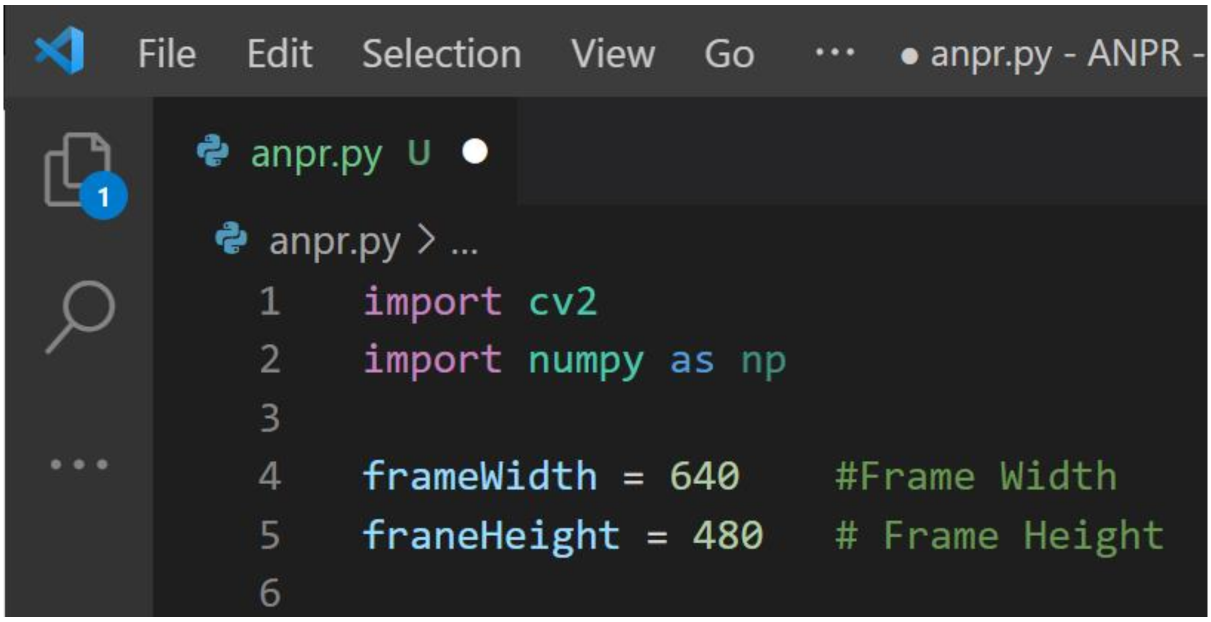
(may be black). The function used is cv2.threshold. First argument is the source image, which should be a grayscale image.

In this project we uses VideoCapture(0) method from cv2 module for using capture live video.



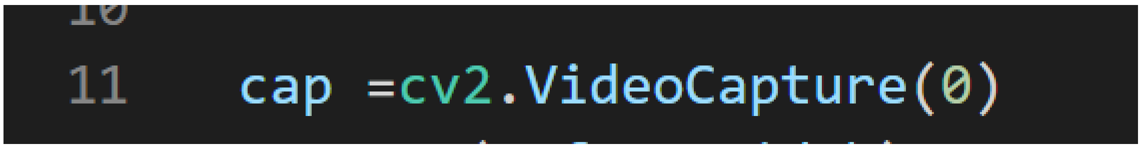
```
import cv2
import numpy as np
```

Here we import cv2 module and numpy module using import statement.



```
import cv2
import numpy as np

frameWidth = 640    #Frame Width
franeHeight = 480   # Frame Height

```

Once importing the required libraries here, we are deciding the frame width and frame height of the Video frame and minimum area of the video frame to 500.



```
cap =cv2.VideoCapture(0)
```

For starting the initial camera of the computer we are going to use VideoCapture() method with argument 0.

For storing the video frames we are using the variable named as **“cap”.** In this statement the 0 is for used to start the initial camera of the computer. If we have external camera connected to the computer then we will use 1 for the external camera.

Haar Cascade XML File

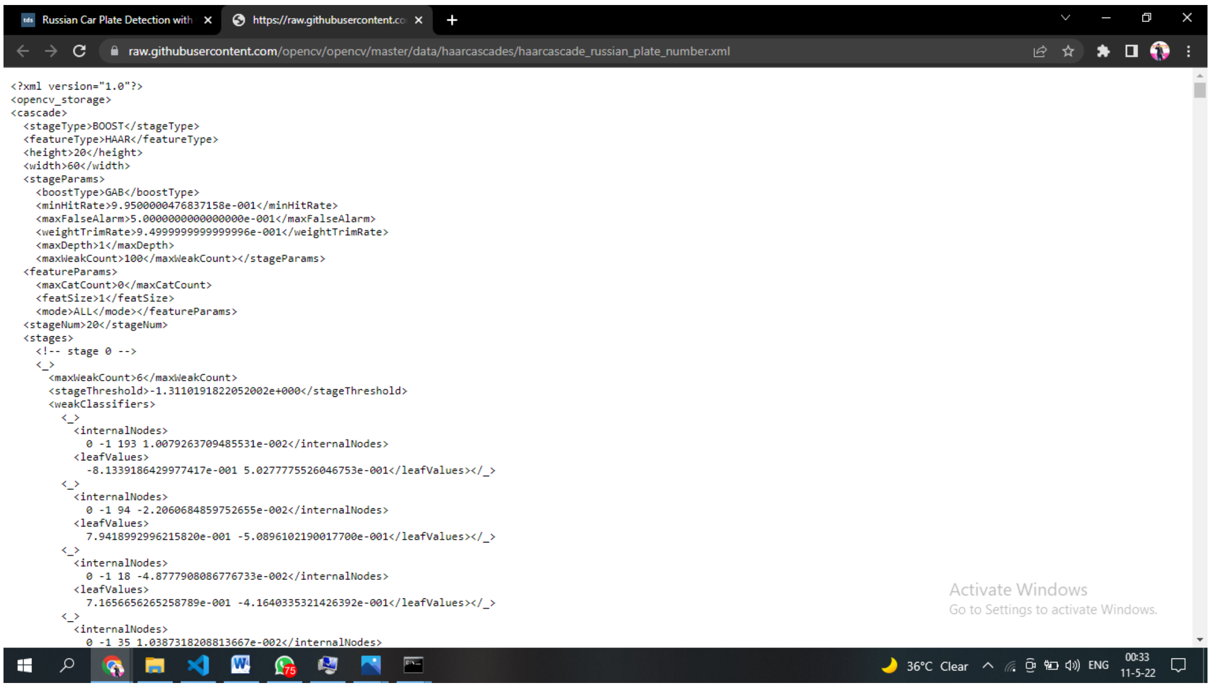
Besides installing the OpenCV library, another important thing to retrieve is the Haar Cascade XML file. Let’s first talk about the theory behind Haar Cascades since it is an important concept. In 2001, Paul Viola and Michael Jones came up with the object detection technique using Haar feature-based cascade classifiers.

It is a machine learning based approach (involving AdaBoost) where a cascade function is trained from many positive and negative images. It extracts numerical values for features (e.g. edges, lines) efficiently with the concept of integral image (or summed-area table), which trumps the default computationally-heavy way of subtracting sums of pixels across multiple regions of an entire image.

In addition, it uses the ‘Cascade of Classifiers’. This means that instead of applying hundreds of classifiers for the many features within the image at one go (which is very inefficient), the classifiers are applied one by one.

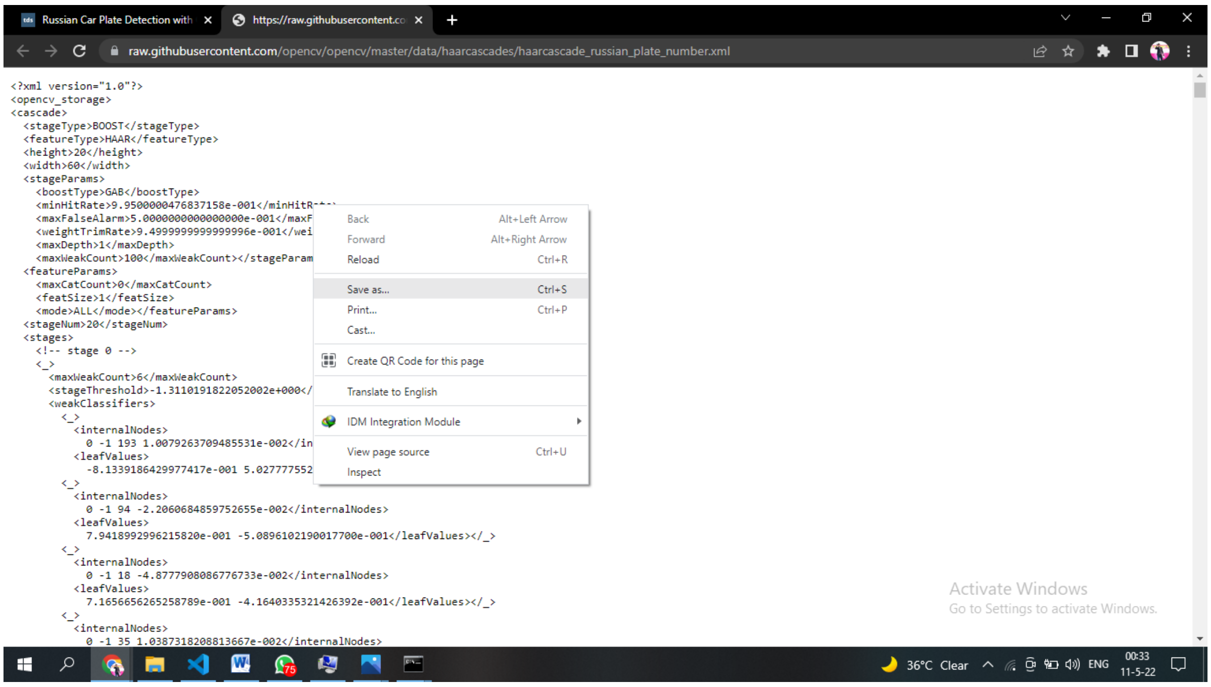
OpenCV actually comes with pre-trained XML files of various Haar Cascades, where each XML file contains the feature set. We will be using the Haar Cascade XML file containing the features for Russian car plates, and here’s how you can download the Haar Cascade XML file by following steps:

1. Visit the OpenCV GitHub page containing the Russian car plate Haar Cascade by clicking here.


Russian Car Plate Detection with
https://raw.githubusercontent.co
raw.githubusercontent.com/opencv/opencv/master/data/haarcascades/haarcascade_russian_plate_number.xml
Activate Windows
Go to Settings to activate Windows.
36°C Clear
ENG
00:33
11-5-22

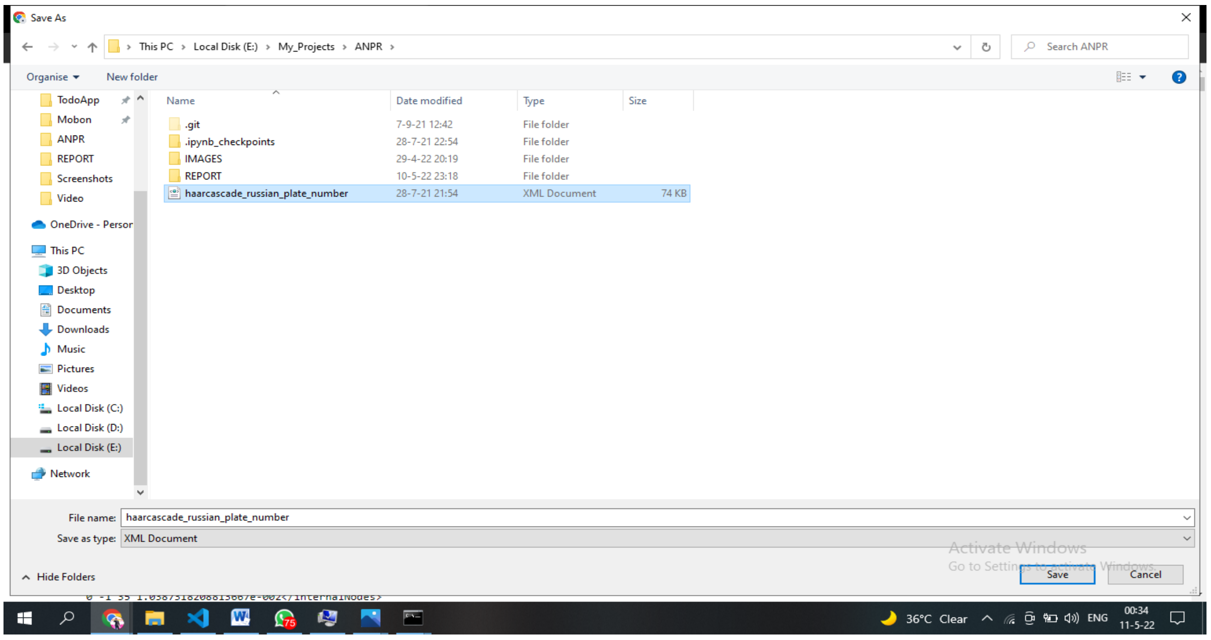
```
<?xml version="1.0"?>
<opencv_storage>
<cascade>
  <stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>20</height>
  <width>60</width>
  <stageParams>
    <boostType>GAB</boostType>
    <minHitRate>9.9500000476837158e-001</minHitRate>
    <maxFalseAlarm>5.0000000000000000e-001</maxFalseAlarm>
    <weightTrimRate>9.4999999999999996e-001</weightTrimRate>
    <maxDepth>1</maxDepth>
    <maxWeakCount>100</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount>
    <featSize>1</featSize>
    <mode>ALL</mode></featureParams>
  <stageNum>20</stageNum>
  <stages>
    <!-- stage 0 -->
    <_>
      <maxWeakCount>6</maxWeakCount>
      <stageThreshold>-1.3110191822052002e+000</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 193 1.0079263709485531e-002</internalNodes>
          <leafValues>
            -8.1339186429977417e-001 5.0277775526046753e-001</leafValues></_>
        <_>
          <internalNodes>
            0 -1 94 -2.2060684859752655e-002</internalNodes>
          <leafValues>
            7.9418992996215820e-001 -5.0896102190017700e-001</leafValues></_>
        <_>
          <internalNodes>
            0 -1 18 -4.8777908086776733e-002</internalNodes>
          <leafValues>
            7.1656656265258789e-001 -4.1640335321426392e-001</leafValues></_>
        <_>
          <internalNodes>
            0 -1 35 1.0387318208813667e-002</internalNodes>
```

2. **Right-click** on the screen (which should be displaying a wall of text with the top line being <?xml version=”1.0"?>), and click ‘**Save as..**’


Russian Car Plate Detection with
https://raw.githubusercontent.co
raw.githubusercontent.com/opencv/opencv/master/data/haarcascades/haarcascade_russian_plate_number.xml
Back Alt+Left Arrow
Forward Alt+Right Arrow
Reload Ctrl+R
Save as... Ctrl+S
Print... Ctrl+P
Cast...
Create QR Code for this page
Translate to English
IDM Integration Module
View page source Ctrl+U
Inspect
Activate Windows
Go to Settings to activate Windows.
36°C Clear
ENG
00:33
11-5-22

```
<?xml version="1.0"?>
<opencv_storage>
<cascade>
  <stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>20</height>
  <width>60</width>
  <stageParams>
    <boostType>GAB</boostType>
    <minHitRate>9.9500000476837158e-001</minHitR
    <maxFalseAlarm>5.0000000000000000e-001</maxF
    <weightTrimRate>9.4999999999999996e-001</wei
    <maxDepth>1</maxDepth>
    <maxWeakCount>100</maxWeakCount></stageParam
  <featureParams>
    <maxCatCount>0</maxCatCount>
    <featSize>1</featSize>
    <mode>ALL</mode></featureParams>
  <stageNum>20</stageNum>
  <stages>
    <!-- stage 0 -->
    <_>
      <maxWeakCount>6</maxWeakCount>
      <stageThreshold>-1.3110191822052002e+000</
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 193 1.0079263709485531e-002</in
          <leafValues>
            -8.1339186429977417e-001 5.027777552
        <_>
          <internalNodes>
            0 -1 94 -2.2060684859752655e-002</internalNodes>
          <leafValues>
            7.9418992996215820e-001 -5.0896102190017700e-001</leafValues></_>
        <_>
          <internalNodes>
            0 -1 18 -4.8777908086776733e-002</internalNodes>
          <leafValues>
            7.1656656265258789e-001 -4.1640335321426392e-001</leafValues></_>
        <_>
          <internalNodes>
            0 -1 35 1.0387318208813667e-002</internalNodes>
```

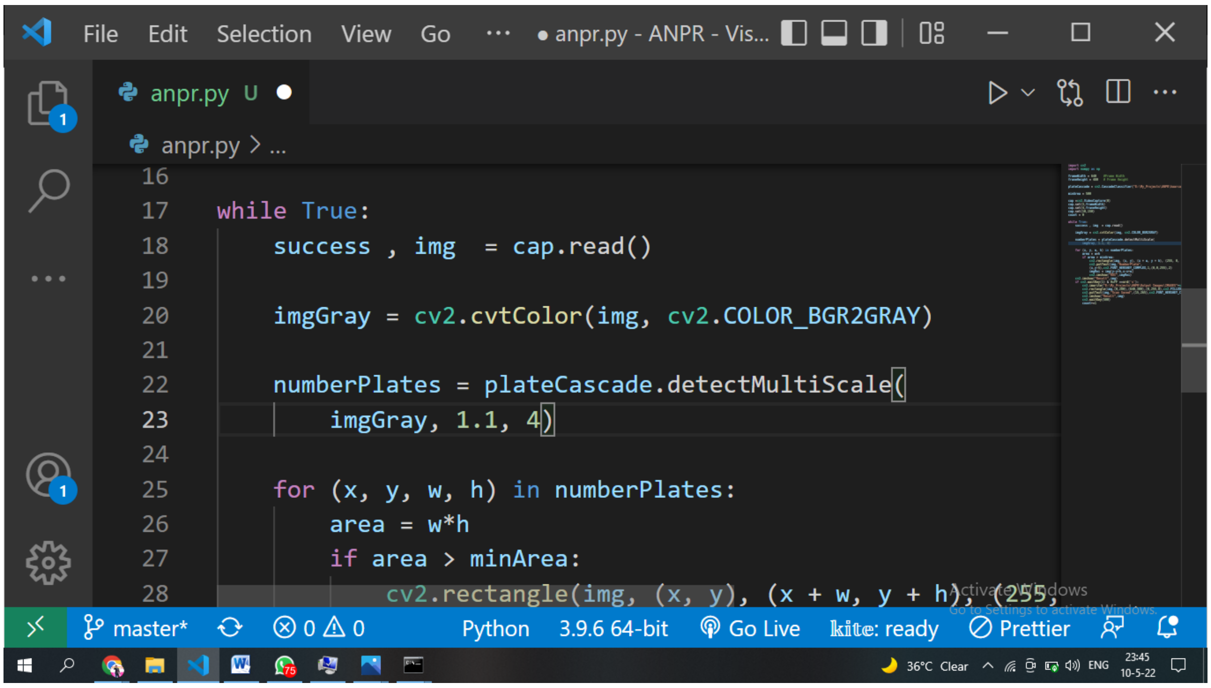
3. In the Save option pop-up, you should see the default file name of '*haarcascade_russian_plate_number*' and file type '*XML document*'. Leave them as the default, and save this XML file in the path where your Jupyter notebook is located.



Name	Date modified	Type	Size
.git	7-9-21 12:42	File folder	
.ipynb_checkpoints	28-7-21 22:54	File folder	
IMAGES	29-4-22 20:19	File folder	
REPORT	10-5-22 23:18	File folder	
haarcascade_russian_plate_number	28-7-21 21:54	XML Document	74 KB

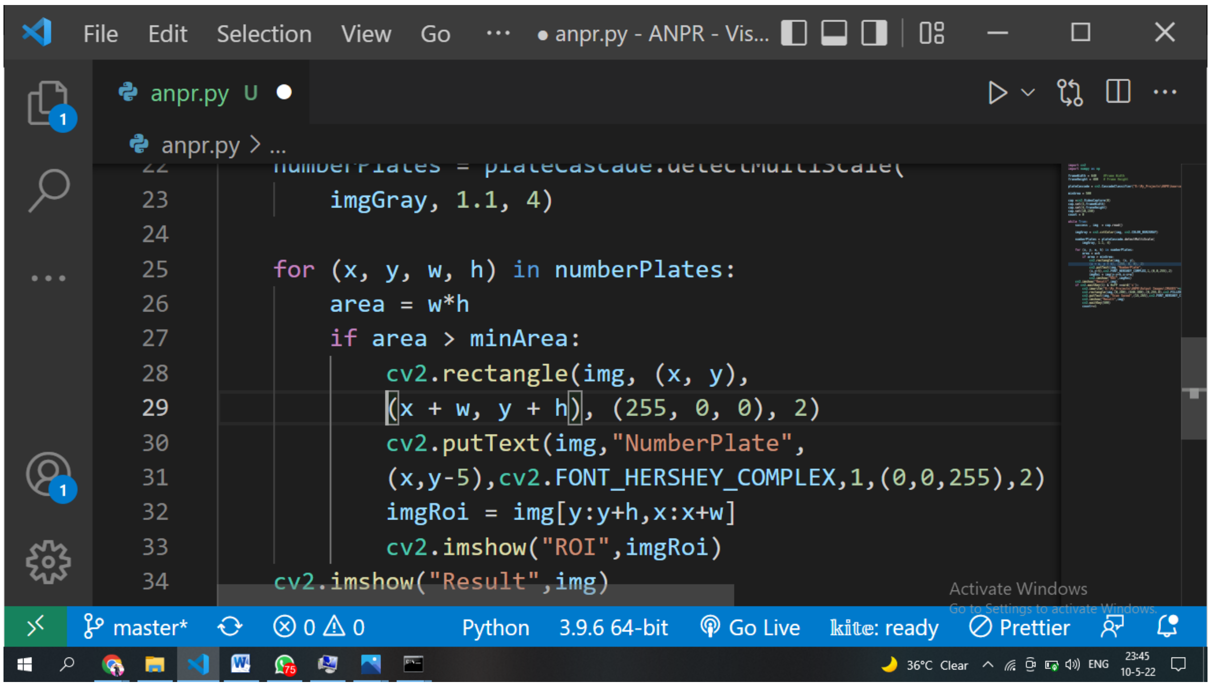
File name: haarcascade_russian_plate_number
Save as type: XML Document

After starting the initial camera of the system and capturing the image or video frame in which the number plate is shown the very first step of the program is convert the color image into gray scale by using function **(cv2.COLOR_BGR2GRAY).**

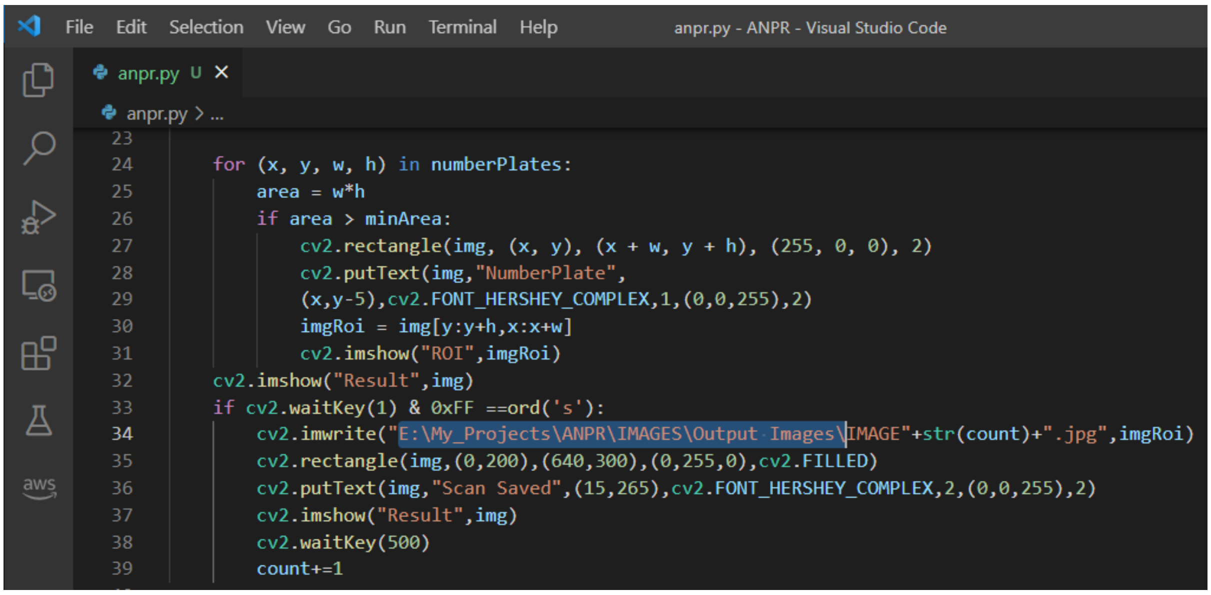

File Edit Selection View Go ··· ● anpr.py - ANPR - Vis...
anpr.py U
anpr.py > ...
master* ⊗ 0 ⚠ 0 Python 3.9.6 64-bit Go Live kite: ready Prettier

```
while True:
    success , img  = cap.read()

    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    numberPlates = plateCascade.detectMultiScale(
        imgGray, 1.1, 4)

    for (x, y, w, h) in numberPlates:
        area = w*h
        if area > minArea:
            cv2.rectangle(img, (x, y), (x + w, y + h), (255,
```

Once converted the input image into gray then it detect the edges of the image by using function **platecascade.detectMultiScale().** For finding the edges we simply use width, and height of our x and y variables in the rectangle format by using function **cv2.rectangle().** And then we storing only the ROI that is Region of Interest of the input image.


File Edit Selection View Go ··· ● anpr.py - ANPR - Vis...
anpr.py U
anpr.py > ...
master* 0 0 Python 3.9.6 64-bit Go Live kite: ready Prettier

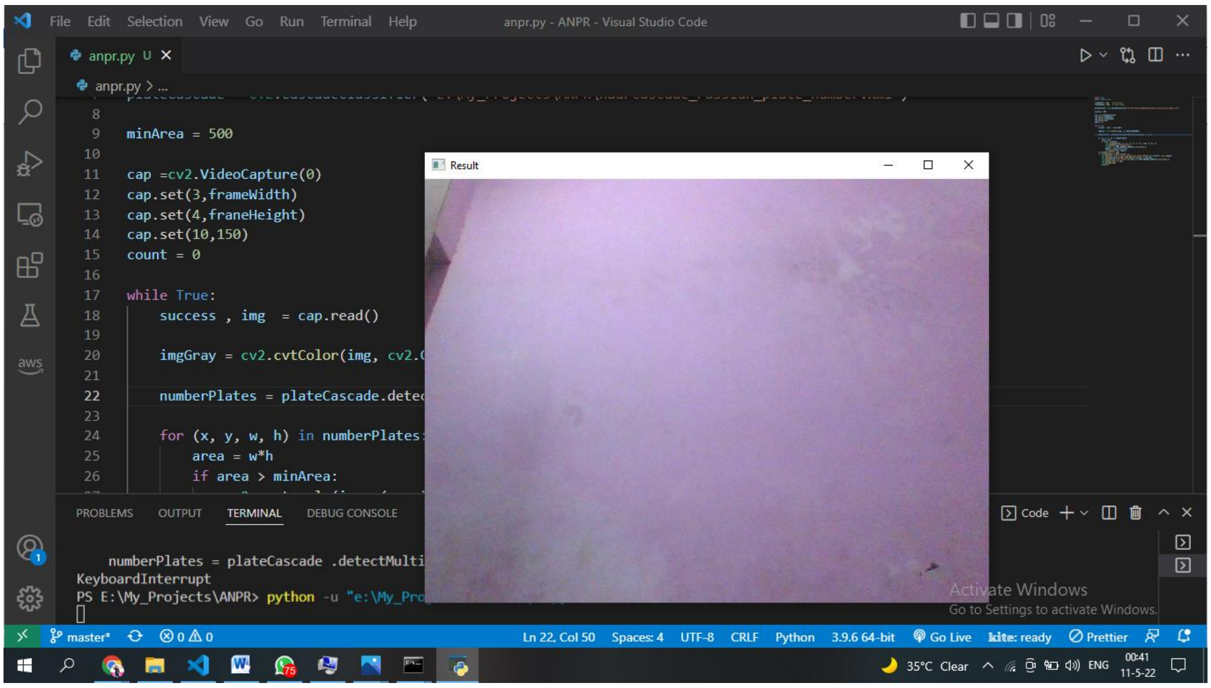
```
    numberPlates = plateCascade.detectMultiScale(
        imgGray, 1.1, 4)

    for (x, y, w, h) in numberPlates:
        area = w*h
        if area > minArea:
            cv2.rectangle(img, (x, y),
            (x + w, y + h), (255, 0, 0), 2)
            cv2.putText(img,"NumberPlate",
            (x,y-5),cv2.FONT_HERSHEY_COMPLEX,1,(0,0,255),2)
            imgRoi = img[y:y+h,x:x+w]
            cv2.imshow("ROI",imgRoi)
    cv2.imshow("Result",img)
```


File Edit Selection View Go Run Terminal Help anpr.py - ANPR - Visual Studio Code
anpr.py U
anpr.py > ...

```

    for (x, y, w, h) in numberPlates:
        area = w*h
        if area > minArea:
            cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
            cv2.putText(img,"NumberPlate",
            (x,y-5),cv2.FONT_HERSHEY_COMPLEX,1,(0,0,255),2)
            imgRoi = img[y:y+h,x:x+w]
            cv2.imshow("ROI",imgRoi)
    cv2.imshow("Result",img)
    if cv2.waitKey(1) & 0xFF ==ord('s'):
        cv2.imwrite("E:\My_Projects\ANPR\IMAGES\Output Images\IMAGE"+str(count)+".jpg",imgRoi)
        cv2.rectangle(img,(0,200),(640,300),(0,255,0),cv2.FILLED)
        cv2.putText(img,"Scan Saved",(15,265),cv2.FONT_HERSHEY_COMPLEX,2,(0,0,255),2)
        cv2.imshow("Result",img)
        cv2.waitKey(500)
        count+=1
```

Screenshots: The screenshots of the implementation of the project is attached below:

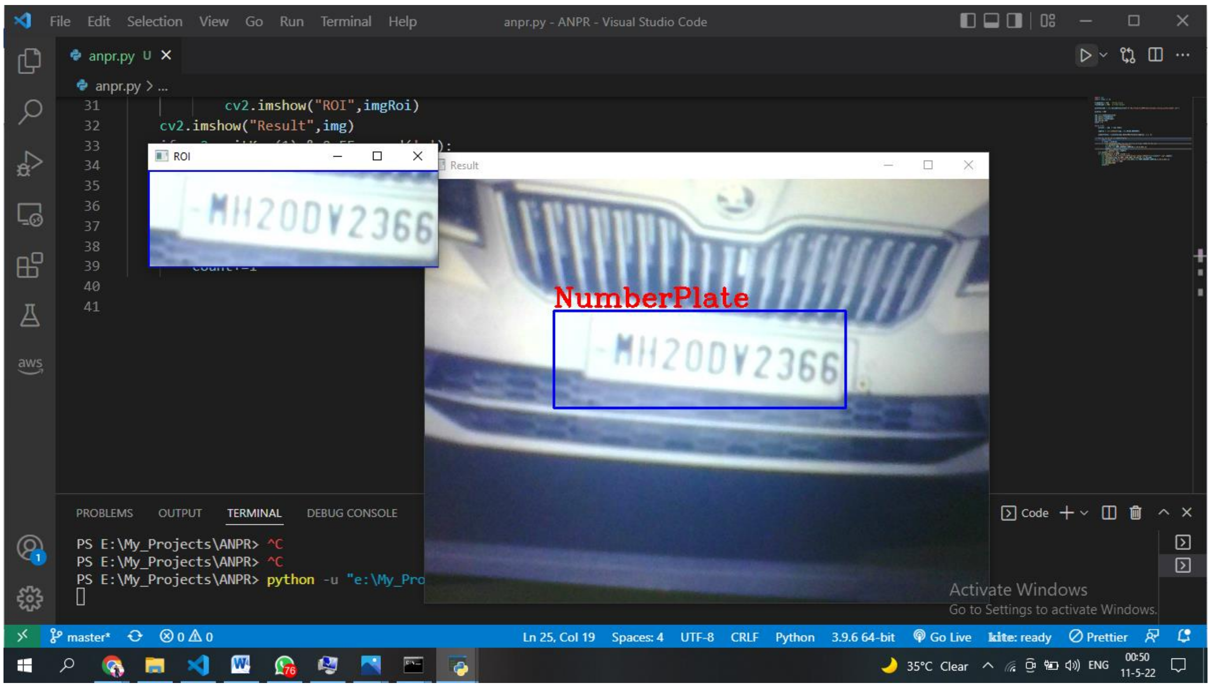
When the code is first run and the initial camera is started, the script performs as:


File Edit Selection View Go Run Terminal Help
anpr.py - ANPR - Visual Studio Code
anpr.py U
anpr.py > ...
Result
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Code
Activate Windows
Go to Settings to activate Windows.
master*
Ln 22, Col 50 Spaces: 4 UTF-8 CRLF Python 3.9.6 64-bit Go Live kite: ready Prettier
35°C Clear ENG 00:41 11-5-22

```
minArea = 500

cap =cv2.VideoCapture(0)
cap.set(3,frameWidth)
cap.set(4,franeHeight)
cap.set(10,150)
count = 0

while True:
    success , img  = cap.read()

    imgGray = cv2.cvtColor(img, cv2.C

    numberPlates = plateCascade.detec

    for (x, y, w, h) in numberPlates:
        area = w*h
        if area > minArea:
```

```
    numberPlates = plateCascade .detectMulti
KeyboardInterrupt
PS E:\My_Projects\ANPR> python -u "e:\My_Pro
```

When the plate is first detected, the script performs as:



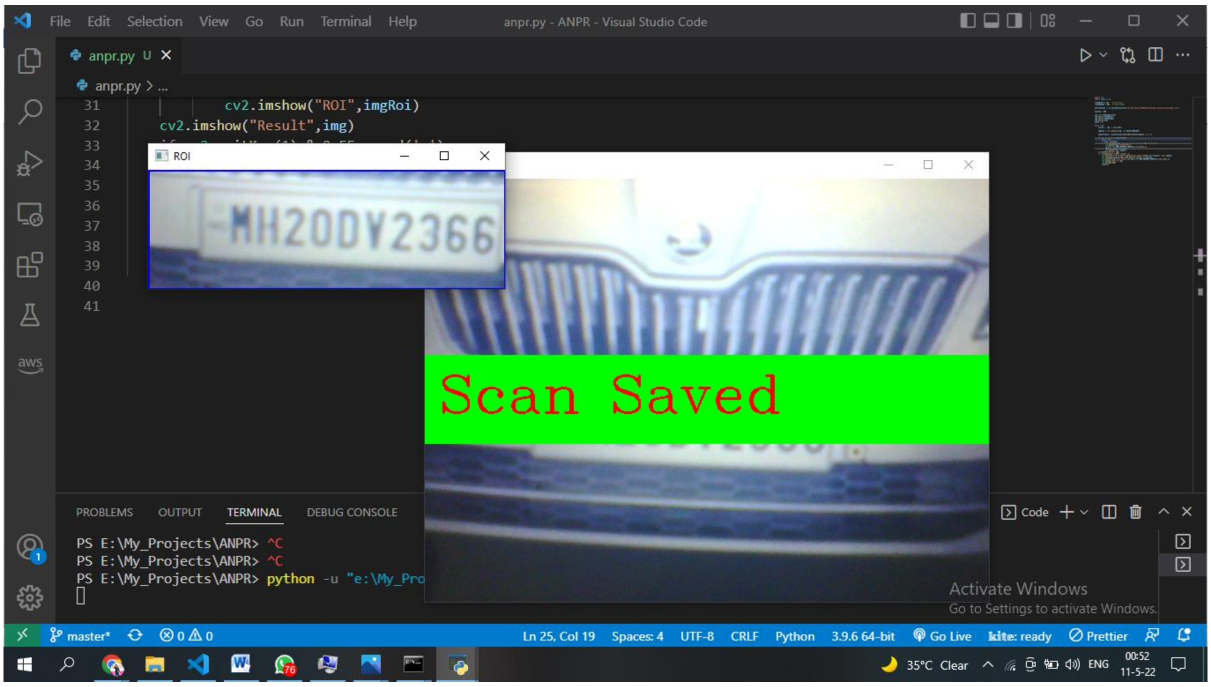
```
            cv2.imshow("ROI",imgRoi)
    cv2.imshow("Result",img)
```

```
PS E:\My_Projects\ANPR> ^C
PS E:\My_Projects\ANPR> ^C
PS E:\My_Projects\ANPR> python -u "e:\My_Pro
```

File Edit Selection View Go Run Terminal Help
anpr.py - ANPR - Visual Studio Code
anpr.py
ROI
Result
NumberPlate
MH20DV2366
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Activate Windows
Go to Settings to activate Windows.
master* Ln 25, Col 19 Spaces: 4 UTF-8 CRLF Python 3.9.6 64-bit Go Live kite: ready Prettier
35°C Clear ENG 00:50 11-5-22

Whenever our proposed system detected the number plate from the given input it will detect the Region of Interest (ROI) in the RIO frame. Whenever the plate is detecting the blue color square box is appeared on the screen with the **NumberPlate** text.

Once plate detected then for saving the scanned images we need to press **“S”** then it saved the scanned image in .jpg format.



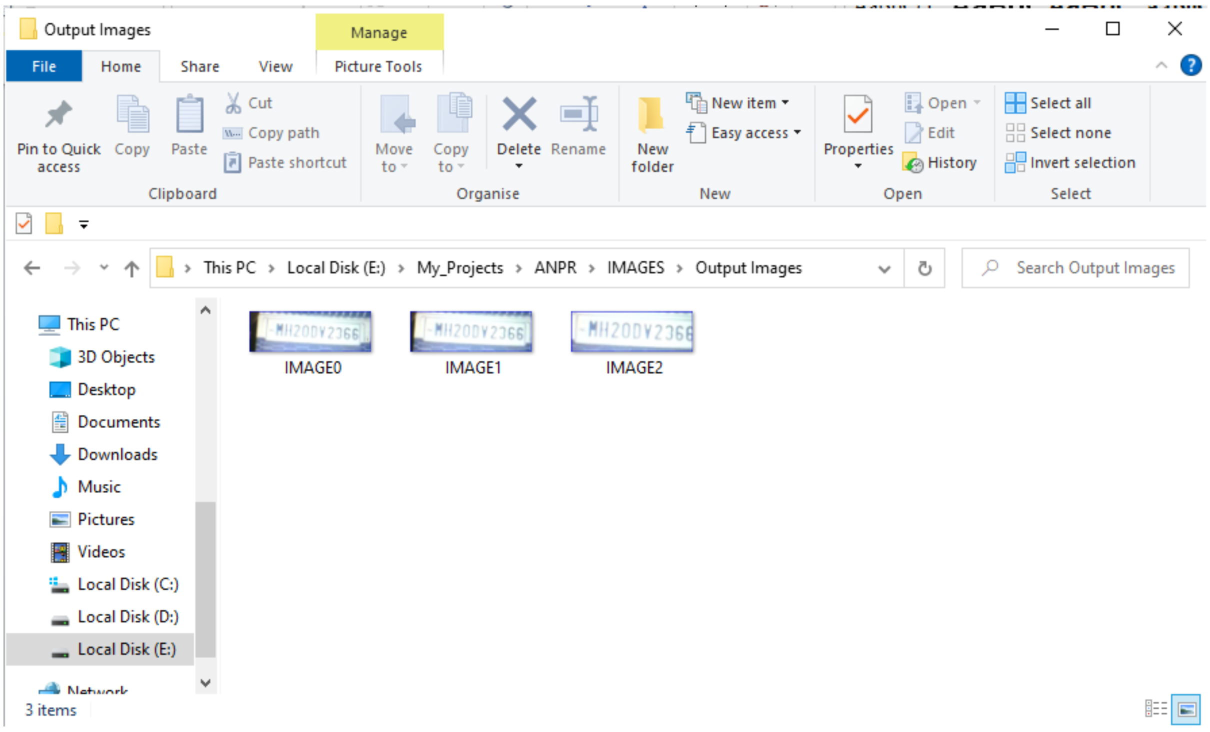
```
            cv2.imshow("ROI",imgRoi)
    cv2.imshow("Result",img)
```

```
PS E:\My_Projects\ANPR> ^C
PS E:\My_Projects\ANPR> ^C
PS E:\My_Projects\ANPR> python -u "e:\My_Pro
```

File Edit Selection View Go Run Terminal Help
anpr.py - ANPR - Visual Studio Code
anpr.py U
ROI
MH20DV2366
Scan Saved
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
master*
Ln 25, Col 19 Spaces: 4 UTF-8 CRLF Python 3.9.6 64-bit Go Live kite: ready Prettier
35°C Clear ENG 00:52 11-5-22

Once plate detected then for saving the scanned images we need to press **“S”** then it saved the scanned image in .jpg format in the desired output location.

Here the location is **E:\My_Projects\ANPR\IMAGES\Output Images**



CHAPTER 7

PROJECT ANALYSIS

7.1 TESTING

- The project script is tested on various pictures and was successful with certain pictures that were available under certain dimension frames.
- The image was successfully enhancing the image quality and was converting the image to binary and morphed image.
- The project extracts the number plate from the car image and displays it separately.
- The project successfully prints half of the License plate number after performing Optical Character Recognition using py-tesseract. The accuracy issues are there in the Tesseract engine and can be enhanced after enhancing the configuration of the engine.

7.2 INFERENCE

The final prototype is performing all the functions including the image acquisition, License plate extraction, pre-processing, character segmentation and character recognition along with printing it on the display or terminal. The project's output images can be stored in the desired location which is provided to the code.

7.3 APPLICATION

Automatic Number Plate Recognition has a wide range of applications since the license number is the primary, most widely accepted, human readable, mandatory identifier of motor vehicles. ANPR provides automated access to the content of the number plate for computer systems managing databases and processing information of vehicle movements. Below we indicated some of the major applications, without the demand of completeness:

- Parking: One of the main applications of ANPR is parking automation and parking.
- Security: ticketless parking fee management, parking access automation, vehicle location guidance, car theft prevention, "lost ticket" fraud, fraud by changing tickets, simplified, partially or fully automated payment process, amongst many others.
- Access Control: Access control in general is a mechanism for limiting access to areas and resources based on users' identities and their membership in various predefined groups. Access to limited zones, however, may also be managed based on the accessing vehicles alone, or together with personal identity. License plate recognition brings automation of vehicle access control management, providing increased security, car pool management for logistics, security guide assistance, event logging, event management, keeping access diary, possibilities for analysis and data mining.
- Border Control: Border Control is an established state-coordinated effort to achieve operational control of the country's state border with the priority mission of supporting the homeland's security against terrorism, illegal cross border traffic, smuggling and criminal activities. Efficient border control significantly decreases the rate of violent crime and increases the society's security. Automatic number plate recognition adds significant value by event logging, establishing investigate-able databases of border crossings, alarming on suspicious passing, at many more.

7.4 ADVANTAGES

- Automatic number plate recognition cameras are used to measure the average vehicle speed over longer distances.
- Used to identify a motorist when he/she drives away without paying for their fuel.
- Targeted advertisement.
- Automatic number plate recognition cameras are used for Traffic management systems.
- Used to analyse the behaviour (route choice, origin-destination etc.) of a motorist for transport planning purposes.
- ANPR camera solutions automatically recognize customers based on their license plate and provide them the complete information about the items that they ordered the last time they used the service.
- Automatic license plate recognition camera solutions are used to recognize the guest vehicles in order to assist visitor management systems.

7.5 DISADVANTAGES

The disadvantage of the license plate recognition system could be privacy rights advocates as well as from citizens concerned with how the cameras are used to track people's movements, and what is done with the data they produce. Others, however, believe the cameras reduce the need for agencies to hire more officers and are the future of law enforcement.

Computerized crime mapping combines geographic information from global positioning satellites with crime statistics collected by a department's CAD system and demographic data provided by private companies or the U.S. Census Bureau.

For example, maps of crimes can be covered with maps or layers of relevant data: unemployment rates in the areas of high crime, locations of abandoned houses, population density, reports of drug activity, or geographic structures (such as alleys, canals, or open fields) that might be contributing factors. Geographic profiling, a moderately new development in the field of environmental criminology analyses the geography of such locations and the sites of the victim encounter, the attack, the murder, and the body dump and maps the most probable location of the suspect's home

CHAPTER 8

CONCLUSION

This project performs mainly four tasks. The first task is to input an image of the car and this will happen with help of the webcam of the computer for the prototype. When the image is fed the image is enhanced in quality. The enhancement is done in the resolution and the thresholding. The image is constraint to a fixed image frame size.

After the enhancement the image is processed to segment the number plate from the full picture based on the mathematical model of the rectangle. The segmented plate is shown in a new window with all the characters in binary form.

The enhanced segmented plate is then processed for OCR or Optical Character Recognition to segment all the characters in the picture in the form of Text and then it can be stored in a database or can be displayed as in this prototype.

The project is designed so that we can understand the technology used in now-a-days Automatic license plate systems and OCR systems used in most of the developed countries like Germany, France, Singapore, Japan, etc.

It is seen that security forces all over the world face problem to locate or register vehicle number to track any culprit. It is also seen that technology can greatly help us in this situation by solving it.

CHAPTER 9

FUTURE SCOPE

As a future work the developed system would be concentrated upon increasing the accuracy of text localization and graphics removal in caption text images. It can be evaluated using various other available image data bases and using various other classifiers.

The proposed methods can be further improvised and applied for automatic mixed mail sorting.

The implementation of the proposed system can be extended for the recognition of number plates of multiple vehicles in a single image frame. User friendly android applications can be developed for traffic surveillance management systems. Also, character recognition can be done using various deep learning algorithms as they yield more accuracy. GPUs can be used to achieve more performance in terms of computational time.

REFERENCES

1. https://www.marketsandmarkets.com/Market-Reports/anpr-system-market-140920103.html
2. https://opensource.google/projects/tesseract
3. https://www.linuxjournal.com/article/9676
4. https://pdfs.semanticscholar.org/bdca/d2b56e3a38ef543f6fb0a602deb5f453493b.pdf?_ga=2.28647292.1514298175.1598968225-502553827.1598968225
5. https://pdfs.semanticscholar.org/4d31/46d2b4bf23558ec0baf93506be5b96437fc2.pdf
6. https://www.researchgate.net/publication/299858935_Proposal_for_Automatic_License_and_Number_Plate_Recognition_System_for_Vehicle_Identification
7. https://opencv.org/#
8. https://pypi.org/project/Pillow/
9. https://www.python.org/about/gettingstarted/
10. https://numpy.org/learn 50
11. G. Liu, Z. Ma, Z. Du and C. Wen, "The calculation method of road travel time based on license plate recognition technology" in Advances in Information Technology and Education, New York, NY, USA:Springer, pp. 385-389, 2011.
12. H. Caner, H. S. Gecim and A. Z. Alkar, "Efficient embedded neural-network-based license plate recognition system", IEEE Trans. Veh. Technol., vol. 57, no. 5, pp. 2675-2683, Sep. 2008.
13. V. Abolghasemi and A. Ahmadyfard, "An edge-based color-aided method for license plate detection", Image Vis. Comput., vol. 27, no. 8, pp. 1134-1142, Jul. 2009.
14. B. Hongliang and L. Changping, "A hybrid license plate extraction method based on edge statistics and morphology", Proc. IEEE 17th ICPR, vol. 2, pp. 831-834, 2004.

15. A. Mousa, "Canny edge-detection based vehicle plate recognition", Int. J. Signal Process. Image Process. Pattern Recognit., vol. 5, no. 3, pp. 1-8, 2012.
16. W. Gao, X. Zhang, L. Yang and H. Liu, "An improved Sobel edge detection", Proc. IEEE 3rd ICCSIT, vol. 5, pp. 67-71, 2010.
17. T. D. Duan, D. A. Duc and T. L. H. Du, "Combining Hough transform and contour algorithm for detecting vehicles' license-plates", Proc. IEEE Int. Symp. Intell. Multimedia Video Speech Process., pp. 747-750, 2004.
18. Amninder Kaur, Sonika Jindal ,Richa Jindal "License Plate Recognition Using Support Vector Machine (SVM)" Dept. Of Computer Science, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 7.
19. ANISH LAZRUS,SIDDHARTHA CHOUBEY,SINHA G.R.,"AN EFFICIENT METHOD OF VEHICLE NUMBER PLATE DETECTION AND RECOGNITION" Department of Computer Science, International Journal of Machine Intelligence, Volume 3, Issue 3.
20. Abhay Singh, Anand Kumar Gupta ,Anmol Singh, Anuj Gupta ,Sherish Johri, "VEHICLE NUMBER PLATE DETECTION USING IMAGE PROCESSING", Department of IT, Volume: 05 Issue: 03 | Mar-2018
21. Ganesh R. Jadhav, Kailash J. Karande, "Automatic Vehicle Number Plate Recognition for Vehicle Parking Management System", IISTE, Vol.5, No.11, 2014.
22. Mutua Simon Mandi ,Bernard Shibwabo, Kaibiru Mutua Raphael, "An Automatic NumberPlate Recognition System for Car Park Management", International Journal of Computer Applications, Volume 175 – No.7, October 2017
23. https://en.wikipedia.org/wiki/Automatic_number-plate_recognition