OBJECT DETECTION: AUTOMATIC LICENSE PLATE DETECTION FOR TRAFFIC CONTROL

AN INTERNSHIP/PROJECT REPORT

Submitted by

CHAUHAN DHRUVILSINH AJAYSINH

200130107112

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

Computer Engineering

Government Engineering College - Gandhinagar

Gujarat Technological University, Ahmedabad

April, 2024

Government Engineering College
Sector-28, Gandhinagar

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **Object Detection: Automatic License Plate Detection For Traffic Control** has been carried out by **Chauhan Dhruvilsinh Ajaysinh** under my guidance in partial fulfillment for the degree of Bachelor of Engineering in Computer Engineering, 8th Semester of Gujarat Technological University, Ahmadabad during the academic year 2023-24.

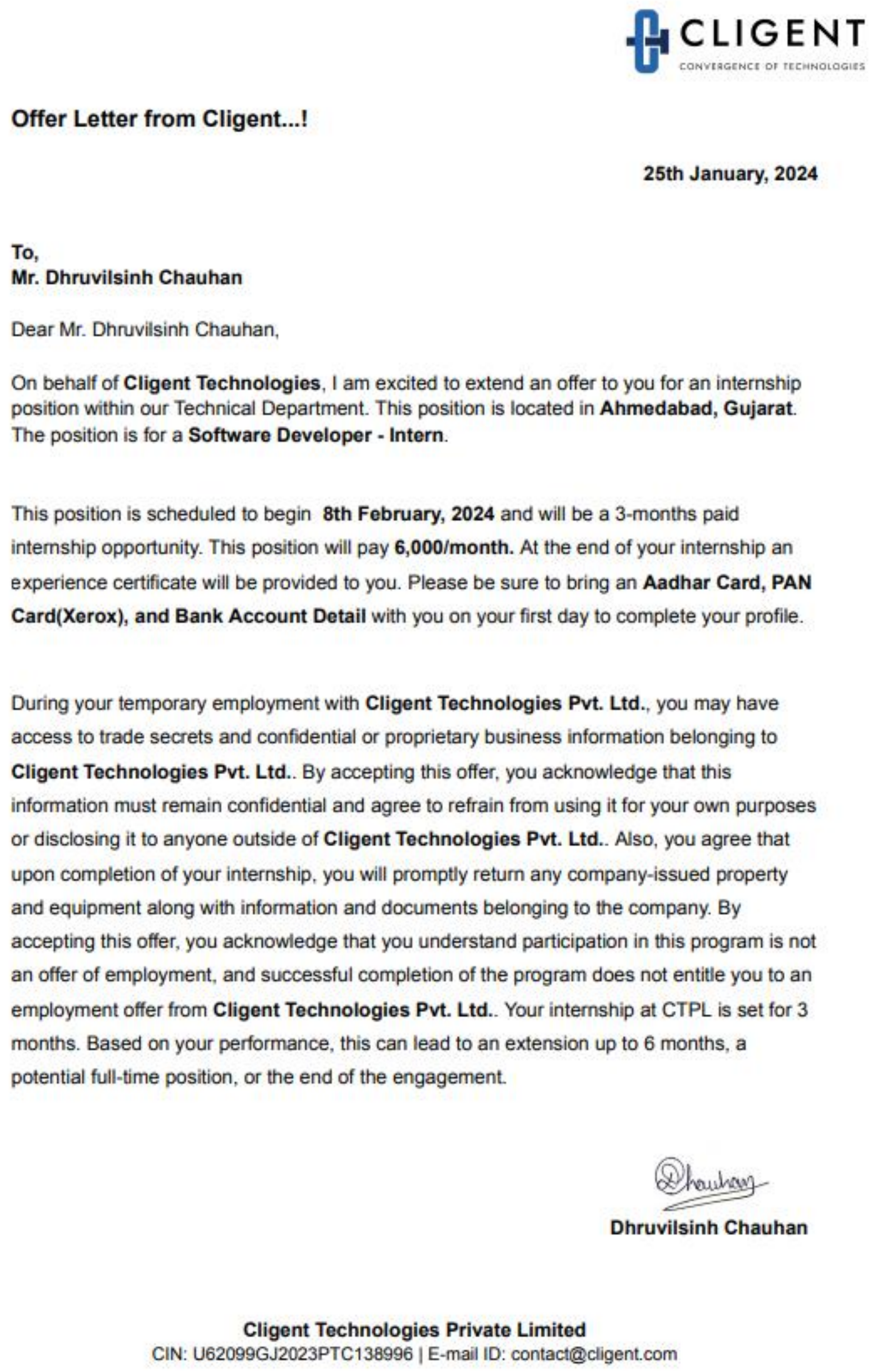
Prof. Viral Patel

Internal Guide

Dr. D. A. Parikh

Head of the Department

OFFER LETTER



Offer Letter from Cligent...!

25th January, 2024

To,
Mr. Dhruvilsinh Chauhan

Dear Mr. Dhruvilsinh Chauhan,

On behalf of **Cligent Technologies**, I am excited to extend an offer to you for an internship position within our Technical Department. This position is located in **Ahmedabad, Gujarat**. The position is for a **Software Developer - Intern**.

This position is scheduled to begin **8th February, 2024** and will be a 3-months paid internship opportunity. This position will pay **6,000/month.** At the end of your internship an experience certificate will be provided to you. Please be sure to bring an **Aadhar Card, PAN Card(Xerox), and Bank Account Detail** with you on your first day to complete your profile.

During your temporary employment with **Cligent Technologies Pvt. Ltd.**, you may have access to trade secrets and confidential or proprietary business information belonging to **Cligent Technologies Pvt. Ltd.**. By accepting this offer, you acknowledge that this information must remain confidential and agree to refrain from using it for your own purposes or disclosing it to anyone outside of **Cligent Technologies Pvt. Ltd.**. Also, you agree that upon completion of your internship, you will promptly return any company-issued property and equipment along with information and documents belonging to the company. By accepting this offer, you acknowledge that you understand participation in this program is not an offer of employment, and successful completion of the program does not entitle you to an employment offer from **Cligent Technologies Pvt. Ltd.**. Your internship at CTPL is set for 3 months. Based on your performance, this can lead to an extension up to 6 months, a potential full-time position, or the end of the engagement.

[illegible]

Dhruvilsinh Chauhan

Cligent Technologies Private Limited
CIN: U62099GJ2023PTC138996 | E-mail ID: contact@cligent.com

CLIGENT
CONVERGENCE OF TECHNOLOGIES

This offer letter represents the full extent of the internship offer and supersedes any prior conversations about the position. Changes to this agreement may only be made in writing. If you have any questions about this offer, please contact Vivek Dhut - One of the Founders at Cligent. We look forward to having you begin your career at **Cligent** and wish you a successful internship.

Welcome to our team!

Sincerely,
From Cligent Technologies,

[illegible]

Harsh Joshi
Director

Dhauhan

Dhruvilsinh Chauhan

Government Engineering College, Gandhinagar

Nr-Animal Vaccine Institute, Sector-28, Gandhinagar, Gujarat 382028

DECLARATION

We hereby declare that the Internship Report submitted along with the Internship entitled **Automatic License Plate Detection For Traffic Control** submitted in partial fulfillment for the degree of Bachelor of Engineering in Computer Engineering to Gujarat Technological University, Ahmedabad, is a bonafide record of original project work carried out by me at **Cligent Technologies** under the supervision of **Mr. Harsh Joshi** and that no part of this report has been directly copied from any students report or taken from any other source, without providing due reference.

Name of the Student	Sign of Student
1. Chauhan Dhruvilsinh Ajaysinh	

Acknowledgement

The satisfaction that accompanied the successful completion of this project would be incomplete without mentioning the people who made it possible, without whose constant guidance and encouragement would have made efforts go in vain. I consider myself privileged to express gratitude and respect towards all those who has gained through the completion of projects.

I convey thanks to my project guide **Mr. Viral Patel**, Computer Engineering department, Government Engineering College, Gandhinagar for providing encouragement, constant support and guidance which was of a great help to complete this project work successfully.

I am grateful to my external guide **Mr. Harsh Joshi**, Co-founder of **Cligent Technologies** for giving me the support and encouragement that was necessary for the completion of this project.

I am grateful to **Dr. Dhaval Parikh**, Head of the Department, Computer Engineering department, Government Engineering College, Gandhinagar for giving us the support and encouragement that was necessary for the completion of this project. I extend my gratitude to all the faculty members for their understanding and guidance that gave me strength to work to long hours for developing a project and preparing the report.

Chauhan Dhruvilsinh Ajaysinh

(200130107112)

Abstract

In recent years the number of vehicles has increased drastically. With this increase, it is becoming difficult to keep track of each vehicle for purpose of law enforcement and traffic management. License Plate Recognition is used increasingly nowadays for automatic toll collection, maintaining traffic activities and law enforcement. Many techniques have been proposed for plate detection, each having its own advantages and disadvantages. The basic step in License Plate Detection is localization of number plate. Automatic License Plate Detection system is a real time software which automatically recognizes the license plate of vehicles. There are many applications ranging from complex security systems to common areas and from parking admission to urban traffic control. Automatic license plate detection has complex characteristics due to diverse effects such as of light and speed. Most of the automatic license plate detection systems are built using proprietary tools like Matlab. This system presents an alternative method of implementing automatic license plate detection systems using free Software including Python and the Open Computer Vision Library.

List of Figures

List of Abbreviations

ABBREVATION		EXPANSION
OCR	-	Optical Character Recognition
CNN	-	Convolutional Neural Network
CV	-	Computer Vision
DL	-	Deep Learning
KNN	-	K-Nearest Neighbor
ALPD	-	Automatic License Plate Detection
LPR	-	License Plate Recognition

Table of Contents

CHAPTER 1: OVERVIEW OF THE COMPANY

Company Name: Cligent Technologies

1.1 The Begining

The Cligent journey began in 2023 when two college friends, Harsh Joshi and Vivek Dhut got together and started with a vision of making world a better place with new technologies for everyone.

1.2 Mission

Cligent is pioneering a unique approach by combining the personalized service of local teams with the strength of a national network, focusing on Engineering, Artificial Intelligence, and the Internet of Things (IoT). Their mission is to lead in these advanced fields, offering expertise and innovative solutions to clients, while staying at the forefront of technological evolution.

1.3 Offerings

AI As A Service:

Their AI experts have experience working on all forms of data ranging from text, number, audio, video to images using various frameworks, data analysis and visualization tools.

IoT Aa A Service:

They design, develop, integrate, and reengineer legacy solutions to accommodate new business objectives with rich experience and expertise in the IoT domain, they help their clients at every stage starting from the hardware design and development to cloud/mobile integration and development.

Software Engineering:

They offer full stack software development services including custom software & application development, cloud-based enterprise & web application development and connected mobile application development.

Data Engineering:

Cligent Technologies data engineering services help companies with highly efficient data engineering solutions with next-gen data analytics and visualization, validating and verifying data quality and implementing ETL processes.

CHAPTER 2: INTRODUCTION

2.1 Introduction

With increasing number of vehicles on roads, it is getting difficult to manually enforce laws and traffic rules for smooth traffic flow. Toll-booths are constructed on freeways and parking structures, where the car has to stop to pay the toll or parking fees. Also, Traffic Management systems are installed on freeways to check for vehicles moving at speeds not permitted by law. All these processes have a scope of improvement. In the centre of all these systems lies a vehicle. In order to automate these processes and make them more effective, a system is required to easily identify a vehicle. The important question here is how to identify a particular vehicle? The obvious answer to this question is by using the vehicle's number plate. 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.

License Plate Recognition (LPR) is a combination of image processing, character segmentation and recognition technologies used to identify vehicles by their license plates. Since only the license plate information is used for identification, this technology requires no additional hardware to be installed on vehicles.

The license plate recognition systems have two main points: the quality of license plate recognition software with recognition algorithms used and the quality of video or image capturing technology, including camera and lighting.

Elements to be considered: maximum recognition accuracy, achieve faster processing speed, handling as many types of plates, manage the broadest range of image/video qualities and achieve maximum distortion tolerance of input data.

CHAPTER 3: OVERVIEW OF PROJECT

3.1 Existing System:

In existing system presents a system called NPR (Number Plate Recognition) which is based on image processing and is used to detect the number plates of vehicles and process them to record the information. In a fast-growing world, it has become almost impossible to track illegal vehicles and store vehicle information. This is eventually leading to a rise in the crime rate, especially due to manual errors. The proposed system first captures the vehicle image and the vehicle plate region is extracted using image segmentation. The resulting data is then used to compare with the records on a database to come up with specific information like vehicle's owner, plate of registration, address, etc. Further, the system is implemented and simulated in MATLAB for studying feasibility and accuracy on a real image.

3.2 Purpose of this project:

- The main purpose of this project is to detect a license plate from a real time video feed by a camera. An efficient algorithm is developed to detect a license plate in a various luminance condition. This algorithm extract license plate from a video using a pre-trained model named haarcascade russian Plate number and provides it as an input to the stage of car licence plate recognition.
- Automatic Number Plate Recognition is a fairly well explored problem with many successful solutions.
- However, these solutions are typically tuned towards a particular environment due to the variations in the features of the number plates across the world.
- Algorithms written for number plate recognition are based on these features and so a universal solution would be difficult to realize as the image analysis techniques that are used to build these algorithms cannot themselves boast hundred percent accuracy.
- The focus of this project is to proposed an algorithm that optimize ALPD process by using highly accurate pre-trained model, haarcascade russian plate number for detecting the license plate of a vehicle. The algorithm is written in python which uses libraries OpenCV, EasyOCR, etc.
- EasyOCR is used to identify the detected characters on the plate.

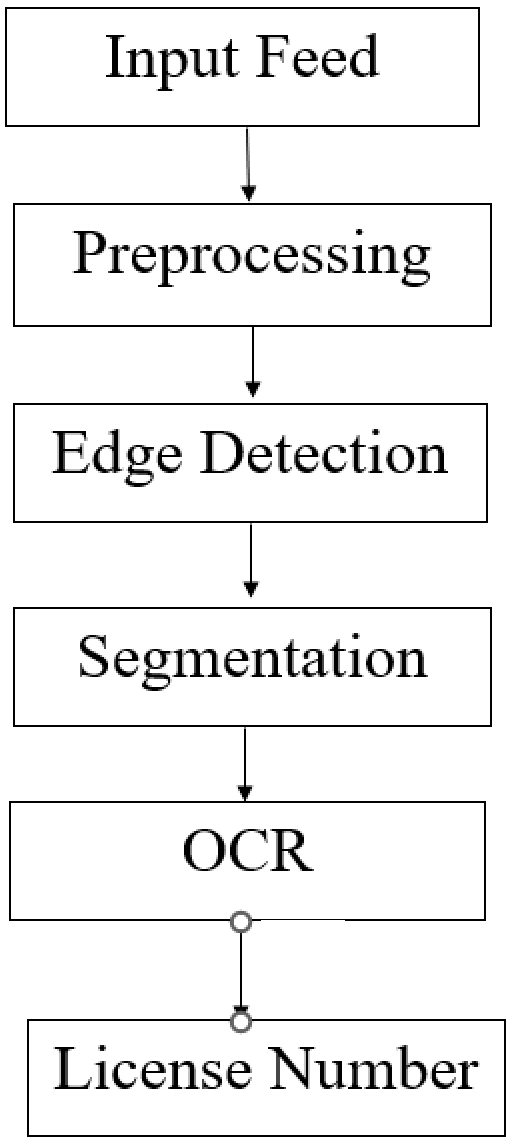


Fig. 3.1 Block Diagram

CHAPTER 4: METHODOLOGY

In this paper the text found on the vehicle plates is detected from the input video feed and this requires the localization of number plate area in order to identify the characters present on it. In literature we can find many methods for number plate detection and recognition system. The major drawback is that how long it will take to compute and recognize the particular license plates. This is critical and most needed when it is applied to real time applications. However, there is always a trade-off between computational time and performance rate. In order to achieve an accurate result and increase the performance of the system more computational time is required. For number plate detection or localization, techniques based on edge statistic and mathematical morphology gives a very good result that uses vertical edge information to calculate the edge density of the image followed by morphology methods such as dilation to extract the region of interest. This technique works well due to the fact that number plates always have high density of vertical edges. But in this method as unwanted edges in the background are also detected which leads to confusion, it is difficult to apply this method for number plates with complex background. We can also use colour based techniques, but the drawback with this method is that it performs well when lighting condition is constant but when there is various illumination condition its performance reduces. But in real-time application normally the images can be obtained with various lighting illumination. Furthermore, the proposed technique is country specific because each country will have different colour code for vehicle number plate. Connected Component Analysis (CCA) method is used to detect the number plate region. CCA is useful for simplifying the detection task. Since it labels binary image into several components based on their connectivity. Based on the problem one can decide on the selection of finding the connected components using 4-adjacency or 8-adjacency of pixels connectivity. Spatial measurement is a measure of spatial characteristics of a connected component such as area, orientation, aspect ratio etc. and filtering is done to eliminate unrelated or unwanted components. When 6 Connected Component Analysis is combined with spatial measurement and filtering produces better result in number plate detection. But for better accuracy and simplicity of model, here I am using a pretrained model, haar cascade russian plate number.xml. Benefits of using pretrained model are that it reduces processing power requirements and provide higher accuracy in number plate detection. Automatic recognition of car license plate number became a very important in our daily life because of the unlimited increase of cars and transportation systems which make it impossible to be fully managed and monitored by humans, examples are so many like traffic monitoring, tracking stolen cars, managing parking toll, red-light violation enforcement, border and customs checkpoints. Yet it’s a very challenging problem, due to the diversity of plate formats, different scales, rotations and non-uniform illumination conditions during image acquisition. This paper presents an approach using simple but efficient morphological operations, filtering and finding techniques for localization of Indian number plates. The algorithm has been tested on 20 samples and is found to extract both alphabets and numbers from vehicle license plates images with an accuracy of 90% for four wheeler license plates of different regions in different climatic conditions. HD number plates are also identified. Ultimately, video is a group of images. So, following are the techniques that are applied on each image inorder to get the desired results.

4.1 PRE-PROCESSING:

4.1.1 Introduction

Image pre-processing is the term for operations on images at the lowest level of abstraction. These operations do not increase image information content but they decrease it if entropy is an information measure. The aim of pre-processing is an improvement of the image data that suppresses undesired distortions or enhances some image features relevant for further processing and analysis task. Image pre-processing use the redundancy in images. Neighbouring pixels corresponding to one real object have the same or similar brightness value. If a distorted pixel can be 98 picked out from the image, it can be restarted as an average value of neighbouring pixels. Image pre-processing methods can be classified into categories according to the size of the pixel neighbourhood that is used for the calculation of a new pixel brightness.

4.2 IMAGE CROPPING AND FILTERING

The first step in image pre-processing is image cropping. Some irrelevant parts of the image can be removed and the image region of interest is focused. This tool provides a user with the size information of the cropped image. Mat Lab function for image cropping realizes this operation interactively waiting for a user to specify the crop rectangle with the mouse and operates on the current axes. The output image is of the same class as the input image. The two-dimensional convolution operation is fundamental to the analysis of images. A new value is ascribed to a given pixel based on the evaluation of a weighted average of pixel values in a k × k neighbourhood of the central pixel. Convolution kernel or the filter mask is represented with weights supplied in a square matrix. It is applied to each pixel in an image. Discrete form of the 2D convolution operator is defined by the following relationship between the elements fi(x, y) of the input image, the elements h(α, β) of the convolution kernel, and the elements g(x, y) of the output image by the following master formula g(x, y) = (k−X 1)/2 α=−(k−1)/2 (k−X 1)/2 β=−(k−1)/2 fi(α, β)h(x − α, y − β), where x, y, α and β are integers [4]. Coefficients of the kernel H represent a discrete approximation of the analytical form of the response function characterizing the desired filter. In practical cases, the kernel is a square array and kx = ky = k, where k is odd and much smaller than the linear image dimension. There are the following steps, realized for each pixel P represented by (x, y):

- Placement of H on P.
- Multiplication of each pixel in the kxk neighbourhood by the appropriate filter mask.
- Summation of all products.
- Placement of the normalized sum into position P of the output image.

This tool for pre-processing lets an user explore 2-D Finite Impulse Response filters. By changing the cut-off frequency and filter order, the user can design filter and can see the designed filter's coefficients and frequency response. Median filtering is a non-linear smoothing method that reduces the blurring of edges and significantly eliminates impulse noise [3, 4]. It suppresses image noise without reducing the image sharpness and can be applied iteratively. The brightness value of the current pixel in the image is replaced by the median brightness of either 3-by-3 or 4-by-4 neighbourhood.

4.3 INTENSITY ADJUSTMENT AND HISTOGRAM EQUALIZATION

A gray-scale transformation T of the original brightness p from scale [p0, pk] into brightness q from a new scale [q0, qk] is given by q = T (p). It does not depend on the position of the pixel in the image. Values below p0 and above pk are clipped. Values below p0 map to q0, and those above pk map to qk. Alpha argument specifies the shape of the curve describing the relationship between the values in the input image and output image. If alpha is less than 1, the mapping is weighted toward brighter output values. If alpha is greater than 1, the mapping is weighted toward lower darker output values. If the argument is omitted its default value is 1. Graphical controls enable a user to increase and decrease the brightness, contrast and alpha correction. Another offered possibility to enhance the contrast of image, by transforming the values in an intensity image so that the histogram of the output image matches 100 a specified histogram, is histogram equalization technique. Region description is based on its statistical grey-level properties. Histogram provides the frequency of the brightness value in the image. An image with n grey levels is represented with one-dimensional array with n elements. The n-th element of array contains the number of pixels whose grey level is n. Assume that the pixel values are normalized and lie in the range [0, 1]. Let s = T(r), for any r ∈ [0, 1], is transformation function which satisfies the following conditions:

- T(r) is single valued and monotonically increasing in the interval [0, 1];
- $0 \leq T(r) \leq 1$ for any r ∈ [0, 1].

The original and transformed grey levels can be characterized by their probability density functions. Contrast is the local change in brightness and is defined as the ratio between average brightness of an object and the background brightness. Histogram equalization technique is based on modifying the appearance of an image by controlling the probability density function of its gray levels by the transformation function T(r). This technique enhances the contrast in the image.

4.4 BRIGHTNESS THRESHOLDING

Brightness thresholding is an indispensable step in extracting pertinent information. A gray-scale image often contains only two level of significant information: the foreground level constituting objects of interest and the background level against which the foreground is discriminated [1]. A complete segmentation of an image R is a finite set of regions R1, R2, Rm, R = [m i=1 Ri ,Ri ∩ Rj = ∅i 6= j. If Rb is a background in the image, then Smi=1, i6=b Ri is considered the object and RC b = Smi=1, i6=b Ri , where RC b is the set complement. While there are two principal 101 peaks of the foreground and the background intensities in the image histogram, there are many other gray intensities present. Binarization can be accomplished by the choice of an intensity, between the two histogram peaks, that is the threshold between all background intensities below and all foreground intensities above. The input image I1 is being transformed to an output binary segmented image I2, in the following way I2(i, j) = (1; I1(i, j) ≥ T 0; I1(i, j) < T where T is the threshold. I2(i, j) = 1 for the object elements and I2(i, j) = 0 for the background elements. There are different approaches for image binarization depending on the type of image [7, 8]. Successful threshold segmentation depends on the threshold selection. A number of conditions like poor image contrast or spatial no uniformities in background intensity can make difficult to resolve foreground from background. These cases require user interaction for specifying the desired object and its distinguishing intensity features.

4.5 CLEARING AREAS OF A BINARY IMAGE

If there is a deformation of the expected shape and size of the border and the whole region during the separation of image object from its background, it can be partially overcome. Usually small polygon mask, located next to the region and out of it, is added to clear image area with similar brightness of the region. This mask can reshape image objects and provides a separation image objects from each other and from their image background. This operation is realized interactively, adding vertices to the polygon. Selecting a final vertex of the polygon over a white colour image region, the fill is started and re-coloured to black. Created fill is a logical mask and the input image is logical matrix. Using logical operator AND under logical arguments, the output image is also obtained as a logical array. If the white regions represents image foreground on the black background, whole objects or their parts can be deleted. Special user requirements about the size and shape of the observed object, can be realized by the same way.

4.6 DETECTING EDGES

Edges are pixels where the intensity image function changes abruptly. Edge detectors are collection of local image pre-processing methods used to locate changes in the brightness function. An image function depends on two variables, co-ordinates in the image plane. Operators describing edges are expressed by partial derivatives. A change of the image function can be described by a gradient that points in the direction of the largest growth of the image function. An edge is a vector variable with two components, magnitude and direction. The edge magnitude is the magnitude of the gradient. The edge direction is rotated with respect to the gradient direction by $-\pi/2$. The gradient direction gives the direction of maximum growth of the function, e.g., from black to white. The boundary and its parts are perpendicular to the direction of the gradient [2, 9]. The gradient magnitude |grad g(x, y)| = s^3 ∂g ∂x ′2 + ^{3}∂g ∂y ′2 and gradient direction ϕ = arg^3 ∂g ∂x, ∂g ∂y ′ are continuous image functions where arg(x, y) is the angle from x axis to the point (x, y). A digital image is descrete in nature and these equations must be approximated by differences of the image g, in the vertical direction for fixed i and in the horizontal direction for fixed j, by following equations Δi g(i, j) = g(i, j) − g(i − n, j) Δj g(i, j) = g(i, j) − g(i, j − n), where n is a small integer chosen so to provide a good approximation to the derivative and to neglect unimportant changes in the image function. 103 Gradient operators approximating derivatives of the image function using differences use one or several convolution masks. Beside them, this tool uses operators based on the zero-crossings of the image function second derivative. Sobel, Prewitt, and Roberts methods find edges by thresholding the gradient and by them horizontal edges, vertical edges or both can be detected. The Laplacian of Gaussian method thresholds the slope of the zero crossings after filtering the image with a LoG filter. Canny method thresholds the gradient using the derivative of a Gaussian filter. One option in the main menu provides processing of the image region of interest or the whole image. It has to move the pointer over the image on the left and when the cursor changes to a crosshair, it has to click on points in the image to select vertices of the region of interest. Offered operations to perform are UN sharping, histogram equalization technique, low pass filtering, median filtering, and brightening, darkening, increasing contrast, decreasing contrast and boundary interpolation.

The pretrained model is used to perform intensity adjustment and histogram equalization, brightness thresholding, clearing areas of binary image and detecting edges which decreases the time and processing complexity.

4.7 FILTER FUNCTION

Filtering in image processing is a process that cleans up appearances and allows for selective highlighting of specific information. A number of techniques are available and the best options can depend on the image and how it will be used. Both analog and digital image processing may require filtering to yield a usable and attractive end result. This can be a routine part of the editing process used to prepare images for distribution.

In the case of film photography, when a photographer develops prints, it may be necessary to use filtering to get the desired effects. Filters can be mounted in the enlarger to improve image quality, or for activities like developing black and white prints from colour negatives. The photographer may perform tests with several filters to find the most appropriate. Film photographers can use filtering in image processing for activities like sharpening up contrast. The filter can change the wavelength of the light as it passes through the enlarger, altering the resulting exposure and developed image. Kits of common filters for enlargers and cameras are widely available commercially.

Digital filtering offers a number of advanced photo manipulation options beyond the basic filters used in photo development. One common use of filtering in image processing is to remove blur. Images may be blurry because of file degradation, moving objects in the frame when the photo was taken, and other issues. The photographer can use a filtration algorithm to selectively target pixels and smooth the image out. More complex filters may be able to reconstruct partially damaged images through averaging, using available data to estimate missing contents in an image. Another use for filtering in image processing is in the handling of images where technicians want to highlight specific objects of interest in the picture. For example, astronomers might pass an image through filters to selectively restrict data from certain wavelengths. This can allow other information in the image to pop into relief. Filters can also remove noise like haze from images to make them cleaner and clearer, even if they are not specifically blurred. Software programs allow for very complex filtering in image processing. Many come with presents that people can use for basic tasks like adding soft filters to portraits or sharpening up contrast in dim images. Users can also develop their own filters, coding in specific parameters to make a custom version for a particular need or project. This may require advanced programming skills, as well as a thorough knowledge of how photography.

CHAPTER 5: SOFTWARE MODULES

IMAGE PROCESSING TOOL: OPENCV

PROGRAM: PYTHON

5.1 PYTHON (PROGRAMMING LANGUAGE)

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library. Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

5.2 HISTORY

Python was conceived in the late 1980s, and its implementation began in December 1989 [28] by Guido van Rossum at Centrum Wiskunde&Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the operating system Amoeba. Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, benevolent dictator for life (BDFL).

About the origin of Python, Van Rossum wrote in 1996: Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus). Python 2.0 was released on 16 October 2000 and had many major new features, including a cycle-detecting garbage collector and support for Unicode. With this release the development process was changed and became more transparent and community-backed. Python 3.0 (which early in its development was commonly referred to as

Python 3000 or py3k), a major, backwards-incompatible release, was released on 3 December 2008 after a long period of testing.

Many of its major features have been backported to the backwards-compatible Python 2.6.xand 2.7.x version series. The End Of Life date (EOL, sunset date) for Python 2.7 was initially set at 2015, then postponed to 2020 out of concern that a large body of existing code cannot easily be forward-ported to Python 3. In January 2017, Google announced work on a Python 2.7 to go transcompiler, which The Register speculated was in response to Python 2.7's planned end-of-life but Google cited performance under concurrent workloads as their only motivation.

Features and philosophy Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and many language features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other 6/13/2017 Python (programming language) paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution.

The design of Python offers some support for functional programming in the Lisp tradition. The language has map(), reduce() and filter() functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The core philosophy of the language is summarized by the document The Zen of Python (PEP 20), which includes aphorisms such as: Beautiful is better than ugly Explicit is better than implicit Simple is better than complex Complex is better than complicated Readability counts Rather than requiring all desired functionality to be built into the language's core, Python was designed to be highly extensible. Python can also be embedded in existing applications that need a programmable interface.

This design of a small core language with a large standard library and an easily extensible interpreter was intended by Van Rossum from the start because of his frustrations with ABC, which espoused the opposite mindset. While offering choice in coding methodology, the Python philosophy rejects exuberant syntax, such as in Perl, in favor of a sparser, less-cluttered grammar.

As Alex Martelli put it: "To describe something as clever is not considered a compliment in the Python culture." Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favor of "there should be one—and preferably only one—obvious way to do it". Python's developers strive to avoid premature optimization, and moreover, reject patches to non-critical parts of CPython that would offer a marginal increase in speed at the cost of clarity.When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or try using PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is making it fun to use. This is reflected in the origin of the name, which comes from Monty Python, and in an occasionally playful approach to tutorials and reference materials, such as using examples that refer to spam and eggs instead of the standard foo and bar. A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style.

To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic. Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonists, Pythonistas, and Pythoneers. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a copy of the entire list. Each element of a slice is a shallow copy. In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality.

For example: List comprehensions vs. for-loops Conditional expressions vs. if blocks The eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements. 6/13/2017 Python (programming language) - Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements.

A particular case of this is that an assignment statement such as a = 1 cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator = for an equality operator == in conditions: if (c = 1) { ... } is syntactically valid (but probably unintended) C code but if c = 1: ... causes a syntax error in Python. Methods Methods on objects are functions attached to the object's class; the syntax instance.method(argument) is, for normal methods and functions, syntactic sugar for Class.method(instance, argument). Python methods have an explicit self parameter to access instance data, in contrast to the implicit self (or this) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby). Typing Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them. Python allows programmers to define their own types using classes, which are most often used for objectoriented programming.

New instances of classes are constructed by calling the class (for example, SpamClass() or EggsClass()), and the classes are instances of the metaclass type (itself an instance of itself), allowing meta programming and reflection. Before version 3.0, Python had two kinds of classes: old-style and new-style. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from object and are instances of type). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0. The long term plan is to support gradual typing and as of Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython. An experimental optional static type checker named mypy supports compile-time type checking.

Mathematics Python has the usual C arithmetic operators (+, -, *, /, %). It also has ** for exponentiation, e.g. 5**3 == 125 and 9**0.5 == 3.0, and a new matrix multiply @ operator is included in version 3.5. [71] Additionally, it has a unary operator (~), which essentially inverts all the bytes of its one argument. For integers, this means ~x=-x-1. Other operators include bitwise shift operators x << y, which shifts x to the left y places, the same as x*(2**y) , and x >> y, which shifts x to the right y places, the same as x/(2**y) . [73] The behavior of division has changed significantly over time: Python 2.1 and earlier use the C division behavior. The / operator is integer division if both operands are integers, and floating-point division otherwise. Integer division rounds towards 0, e.g. 7 / 3 == 2 and -7 / 3 == -2. Python 2.2 changes integer division to round towards negative infinity, e.g. 7 / 3 == 2 and -7 / 3 == -3. The floor division // operator is introduced. So 7 // 3 == 2, -7 // 3 == -3, 7.5 // 3 == 2.0 and 6/13/2017 Python (programming language) – Wikipedia https://en.wikipedia.org/wiki/Python_(programming_language) 8/19 -7.5 // 3 == -3.0.

Adding from __future__ import division causes a module to use Python 3.0 rules for division (see next). Python 3.0 changes / to be always floating-point division. In Python terms, the pre-3.0 / is classic division, the version-3.0 / is real division, and // is floor division. Rounding towards negative infinity, though different from most languages, adds consistency. For instance, it means that the equation (a+b) // b == a // b + 1 is always true. It also means that the equation b * (a // b) + a % b == a is valid for both positive and negative values of a. However, maintaining the validity of this equation means that while the result of a % b is, as expected, in the half-open interval [0, b), where b is a positive integer, it has to lie in the interval (b, 0] when b is negative. Python provides a round function for rounding a float to the nearest integer. For tie-breaking, versions before 3 use round-away-from-zero: round(0.5) is 1.0, round(-0.5) is −1.0. Python 3 uses round to even: round(1.5) is 2, round(2.5) is 2. Python allows boolean expressions with multiple equality relations in a manner that is consistent with general use in mathematics.

For example, the expression a < b < c tests whether a is less than b and b is less than c. Cderived languages interpret this expression differently: in C, the expression would first evaluate a < b, resulting in 0 or 1, and that result would then be compared with c. Python has extensive built-in support for arbitrary precision arithmetic. Integers are transparently switched from the machine-supported maximum fixed-precision (usually 32 or 64 bits), belonging to the python type int, to arbitrary precision, belonging to the python type long, where needed. The latter have an "L" suffix in their textual representation. (In Python 3, the distinction between the int and long types was eliminated; this behavior is now entirely contained by the int class.) The Decimal type/class in module decimal (since version 2.4) provides decimal floating point numbers to arbitrary precision and several rounding modes.The Fraction type in module fractions (since version 2.6) provides arbitrary precision for rational numbers.

Due to Python's extensive mathematics library, and the third-party library NumPy which further extends the native capabilities, it is frequently used as a scientific scripting language to aid in problems such as numerical data processing and manipulation. Libraries Python has a large standard library, commonly cited as one of Python's greatest strengths, providing tools suited to many tasks. This is deliberate and has been described as a "batteries included" Python philosophy.

5.3 OPENCV

OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-sourceBSD license.

OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe.

5.3.1 History:

Officially launched in 1999 the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time tracing and 3D display walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were described as:

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

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. A version 1.1 "pre-release" was released in October 2008. The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months] and development is now done by an independent Russian team supported by commercial corporations. In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site. On May 2016, Intel signed an agreement to acquire Itseez, a leading developer of OpenCV.

5.3.2 Applications:

OpenCV's application areas include:

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human-computer interaction (HCI)
- Mobile robotics
- Motion robotics
- Motion robotics
- Motion understanding
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Motion tracking
- Augmented reality

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- Boosting
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- K-nearest neighbour algorithm
- Naïve Bayes classifier
- Artificial neural network
- Random forest
- Support vector machine (SVM)
- Deep neural networks (DNN)

5.4 PROGRAMMING LANGUAGE

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

Since version 3.4, OpenCV.js is a JavaScript binding for selected subset of OpenCV functions for the web platform.

All of the new developments and algorithms in OpenCV are now developed in the C++ interface.

HARDWARE ACCELERATION

If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself.

A CUDA-based GPU interface has been in progress since September 2010.

An OpenCL-based GPU interface has been in progress since October 2012, documentation for version 2.4.13.3 can be found at docs.opencv.org.

5.4.1 OS Support

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.

5.5 WHAT IS OCR?

OCR is formerly known as Optical Character Recognition which is revolutionary for the digital world nowadays. OCR is actually a complete process under which the images/documents which are present in a digital world are processed and from the text are being processed out as normal editable text.

5.5.1 Purpose Of OCR

OCR is a technology that enables you to convert different types of documents, such as scanned paper documents, PDF files, or images captured by a digital camera into editable and searchable data.

5.5.2 What is EasyOCR?

EasyOCR is actually a python package that holds PyTorch as a backend handler. EasyOCR like any other OCR(tesseract of Google or any other) detects the text from images but in my reference, while using it I found that it is the most straightforward way to detect text from images also when high end deep learning library(PyTorch) is supporting it in the backend which makes it accuracy more credible. EasyOCR supports 42+ languages for detection purposes. EasyOCR is created by the company named Jaided AI company.

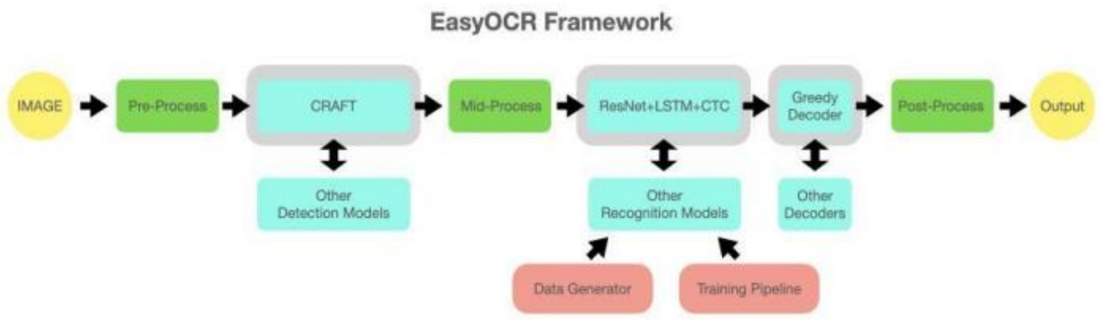
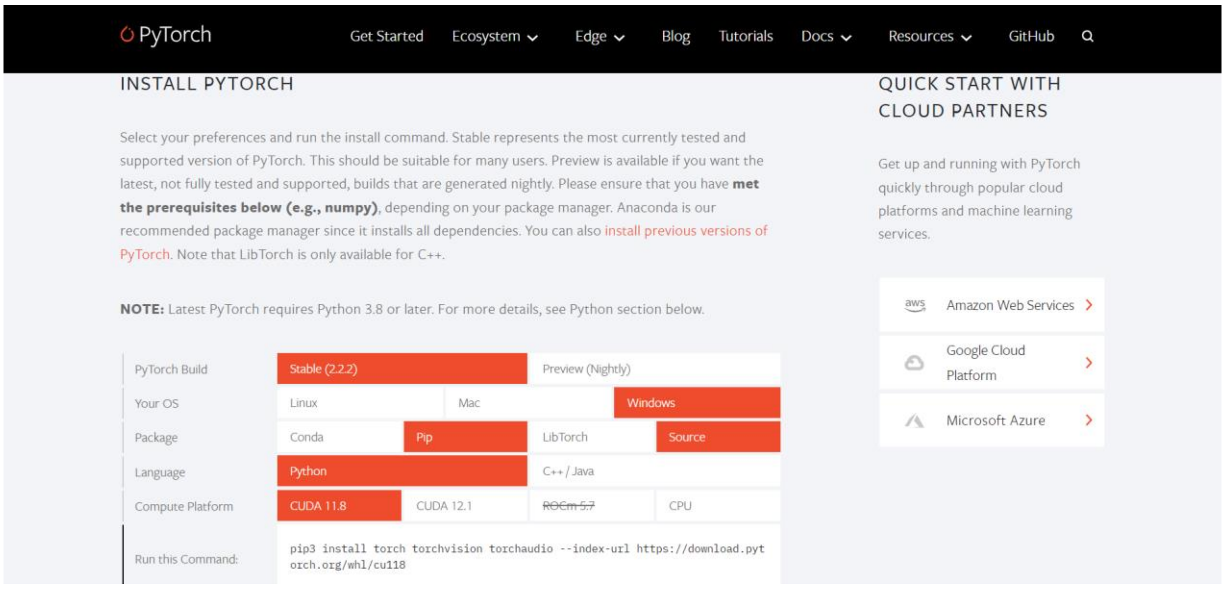


Fig. 5.1 EasyOCR Framework

Table of content

1. Install core dependencies

- **Pytorch:**
  Installing PyTorch as a complete package can be a little tricky so I would recommend traversing through the official site of PyTorch. When you will open its official site then that's what you will see in its interface as in the image below.



INSTALL PYTORCH

Select your preferences and run the install command. Stable represents the most currently tested and supported version of PyTorch. This should be suitable for many users. Preview is available if you want the latest, not fully tested and supported, builds that are generated nightly. Please ensure that you have **met the prerequisites below (e.g., numpy)**, depending on your package manager. Anaconda is our recommended package manager since it installs all dependencies. You can also install previous versions of PyTorch. Note that LibTorch is only available for C++.

NOTE: Latest PyTorch requires Python 3.8 or later. For more details, see Python section below.

PyTorch Build	[✓ Stable (2.2.2)]	Preview (Nightly)		
Your OS	Linux	Mac	[✓ Windows]	
Package	Conda	[✓ Pip]	LibTorch	Source
Language	[✓ Python]	C++ / Java		
Compute Platform	[✓ CUDA 11.8]	CUDA 12.1	~~ROCm 5.7~~	CPU

Run this Command:

```
pip3 install torch torchvision torchaudio --index-url https://download.pyt
orch.org/whl/cu118
```

QUICK START WITH CLOUD PARTNERS

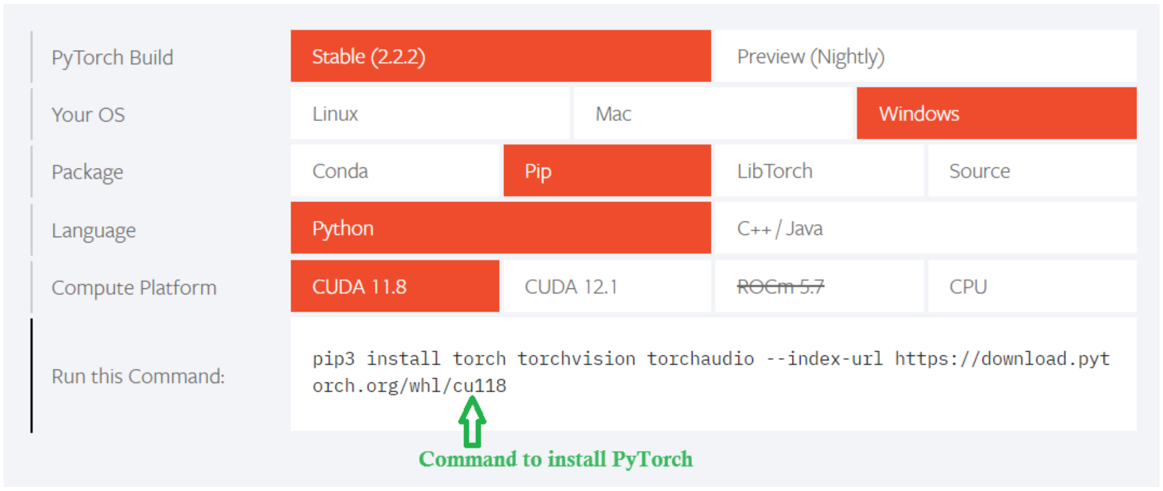
Get up and running with PyTorch quickly through popular cloud platforms and machine learning services.

- Amazon Web Services
- Google Cloud Platform
- Microsoft Azure

Fig. 5.2 PyTorch

Image Source: PyTorch

Now, if you will look closely at the above image one can find out that there are numerous options available for us to choose from and get the command most compatible according to our choices.



PyTorch Build	[✓ Stable (2.2.2)]	Preview (Nightly)		
Your OS	Linux	Mac	[✓ Windows]	
Package	Conda	[✓ Pip]	LibTorch	Source
Language	[✓ Python]	C++ / Java		
Compute Platform	[✓ CUDA 11.8]	CUDA 12.1	~~ROCm 5.7~~	CPU

Run this Command:

```
pip3 install torch torchvision torchaudio --index-url https://download.pyt
orch.org/whl/cu118
```

Fig 5.3 Installing PyTorch

Image Source: PyTorch

In the above representation, one can notice that I have chosen the Package: pip and Compute platform: CPU and based on my choices I got the command as – pip install torch torchvisiontorchaudio. After getting this command it would be like walking on a cake, simply just run this command on your command prompt and your PyTorch library will be installed successfully.

- EasyOCR

After installing the PyTorch library successfully it’s quite easy to install the EasyOCR library, one just has to run the following command:

```
pip install easyocr
```

CHAPTER 6: RESULTS AND DISCUSSION

Video is a group of frames/images. Following are the steps that are performed on each image to fetch the accurate results.

6.1 Gray Image Conversion

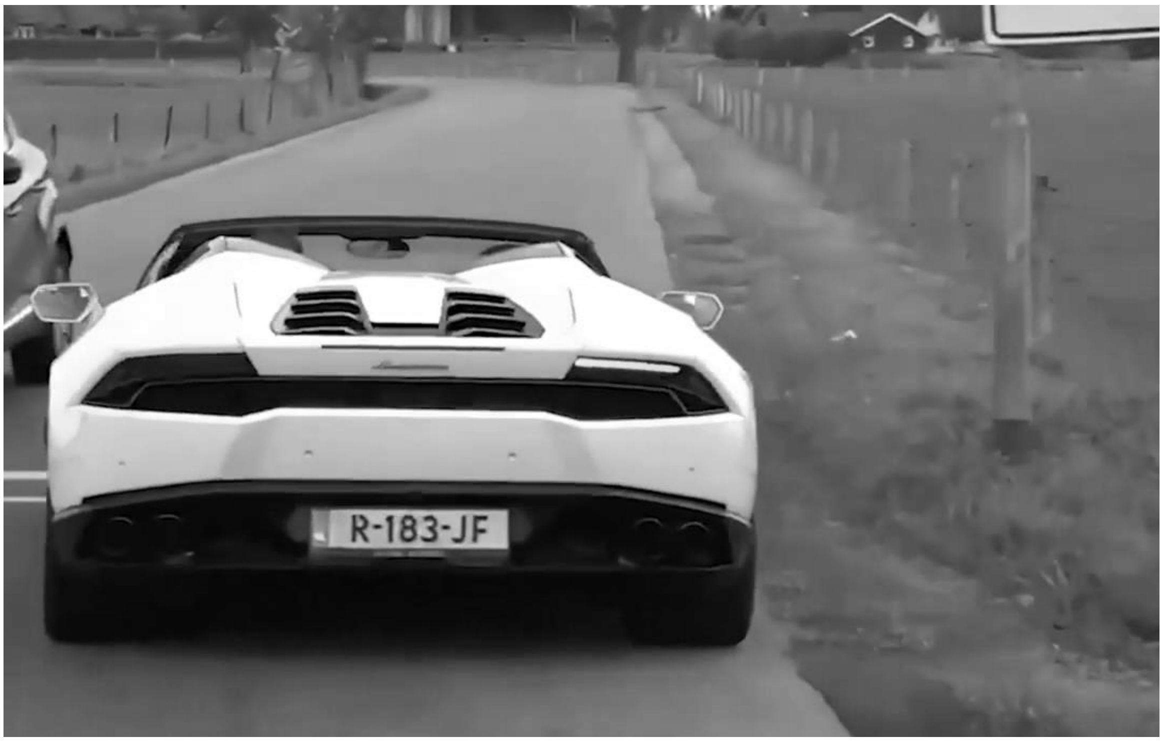


Fig 6.1 Gray Image

In this step, firstly I am converting the BGR colour image into grey image. Grey image makes the detection process easy and accurate.

6.2 License Plate Detection

We are extracting the region of interest from the grey image.

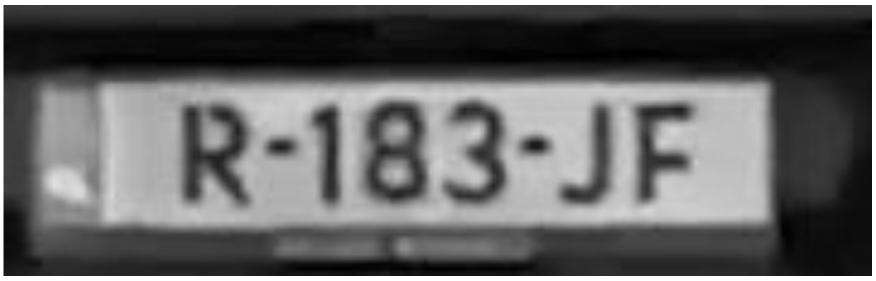


Fig 6.2 Detecting License Plate

6.3 License Plate Number Extraction

EasyOCR is the open source python library which helps us to extract the numbers and characters from the image. Following are the characters and numbers extracted from the image:

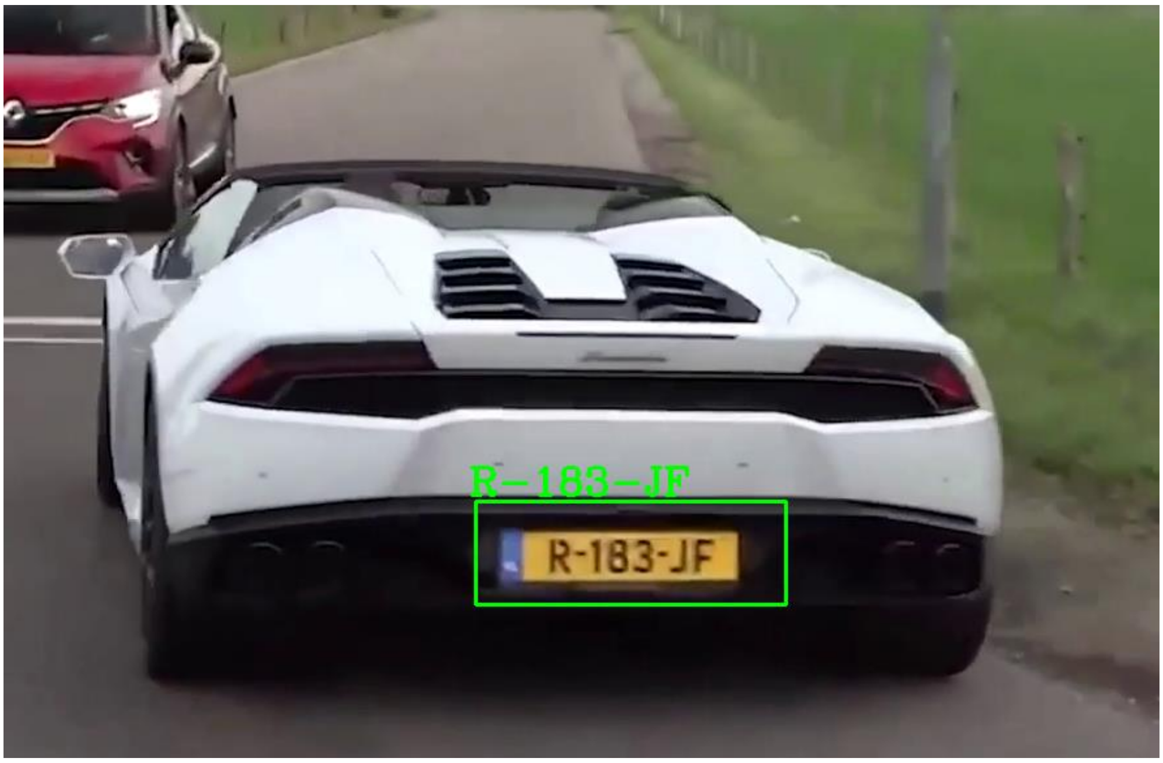
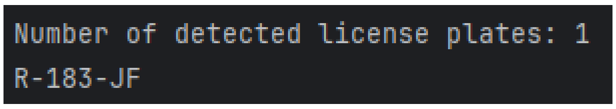


Fig 6.3 Extracting the License plate unique number

6.4 Extracted Number

The extracted number is printed in the console/terminal. We can store this number in database as well.



```
Number of detected license plates: 1
R-183-JF
```

Fig 6.4 Detected Number

The pre-trained model makes our work fast and easy.

CHAPTER 7: CONCLUSION AND FUTURE WORK

License Plate Recognition was a computer system that recognizes any digital image automatically on the number plate. This system includes various operations such as taking pictures, localizing the number pad, truncating characters and OCR from alphanumeric characters. The main conclusion 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 and to identify each character of the segment by using the Open Computer Vision Library. This has been implemented in CNN 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.

REFERENCE:

[1] Martin, F., Garcia, M., Alba, J.L. New methods for automatic reading of VLP's (Vehicle License Plates) (2002) Proc. IASTED Int. Conf. SPPRA,

[2] Hongliang, B., Changping, L. A hybrid license plate extraction method based on edge statistics and morphology (2004) Proc. ICPR, pp. 831-834.

[3] Zheng, D., Zhao, Y., Wang, J. An efficient method of license plate location (2005) Pattern Recognit. Lett, 26 (15), pp. 2431-2438. Nov

[4] Wang, S., Lee, H. Detection and recognition of license plate characters with different appearences (2003) Proc. Conf. Intell. Transp. Syst, 2, pp. 979-984.

[5] Lee, H.-J., Chen, S.-Y., Wang, S.-Z. Extraction and recognition of license plates of motorcycles and vehicles on highways (2004) Proc. ICPR, pp. 356-359.

[6] Comelli, P., Ferragina, P., Granieri, M.N., Stabile, F. Optical recognition of motor vehicle license plates (1995) IEEE Trans. Veh. Technol, 44 (4), pp. 790-799. Nov

[7] Shi, X., Zhao, W., Shen, Y. (2005) Automatic License Plate Recognition System Based on Color Image Processing, 3483, pp. 1159-1168. O. Gervasi et al, Ed. New York: Springer-Verlag

[8] Kim, K.I., Jung, K., Kim, J.H. Color Texture-Based Object Detection: An Application to License Plate Localization, 2388, pp. 293-309. S.-W. Lee and A. Verri, Eds. New York: Springer-Verlag, pp

[9] Zhong, Y., Jain, A.K. Object localization using color, texture and shape (2000) Pattern Recognit, 33 (4), pp. 671-684. Apr

[10] Suryanarayana, P.V., Mitra, S.K., Baneree, A., Roy, A.K. A morphology based approach for car license plate extraction (2005) Proc. IEEE Indicon, pp. 24-27.Chennai, India, Dec

SOURCE CODE:

1. Using pre-trained model: (haar cascade russian plate number.xml)

```
# import required libraries
import cv2
import numpy as np
import easyocr

# Read input image
cap = cv2.VideoCapture("s.mp4") # for video
# cap = cv2.imread("car.jpg")  #for image

reader = easyocr.Reader(['en'])
while True:
    success, img = cap.read()

    # convert input image to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # read haarcascade for number plate detection
    cascade = cv2.CascadeClassifier('haarcascade_russian_plate_number
(1).xml')

    # Detect license number plates
    plates = cascade.detectMultiScale(gray, 1.2, 5)
    print('Number of detected license plates:', len(plates))

    # loop over all plates
    for (x, y, w, h) in plates:
        # draw bounding rectangle around the license number plate
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
        gray_plates = gray[y:y + h, x:x + w]
        color_plates = img[y:y + h, x:x + w]
        # reader = easyocr.Reader(['en'])
        gray_plate_final = cv2.cvtColor(gray_plates, cv2.COLOR_BGR2GRAY)
        #_, license_plate_threshold = cv2.threshold(gray_plate_final, 1,
255, cv2.THRESH_BINARY_INV)
        output = reader.readtext(gray_plate_final)
        for out in output:
            txt_bbox, text, txt_score = out
            if txt_score > 0.01:
                cv2.putText(img,text.upper(),(x-5,y-
5),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
                print(text.upper())

        # save number plate detected
        cv2.imwrite('Numberplate.jpg', gray_plate_final)
        cv2.imshow('Number Plate', gray_plate_final)
        cv2.imshow('Number Plate Image', img)
        cv2.waitKey(1)
```

2. Without using pretrained model:

```
3. import cv2
   import numpy as np
   import easyocr
```

```python
def detect_and_read_plate(img_path):
    reader = easyocr.Reader(['en'], gpu=False)

    img = cv2.imread(img_path)
    if img is None:
        print("Error: Image could not be read.")
        return

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

    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    # Edge detection
    edged = cv2.Canny(blurred, 30, 200)
    # Find contours and sort them by size
    contours, _ = cv2.findContours(edged.copy(), cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]

    # Screen for potential number plates
    candidate_number_plate = None
    for contour in contours:
        # Approximate the contour
        peri = cv2.arcLength(contour, True)
        approx = cv2.approxPolyDP(contour, 0.02 * peri, True)

        # Contours with 4 points can potentially be the number plate
        if len(approx) == 4:
            candidate_number_plate = approx
            break

    if candidate_number_plate is not None:
        # Extract the number plate from the image
        x, y, w, h = cv2.boundingRect(candidate_number_plate)
        plate_img = img[y:y + h, x:x + w]

        # Use EasyOCR to read the text from the number plate
        result = reader.readtext(plate_img)
        plate_text = ' '.join([text[1] for text in result])
        print("Detected Number Plate:", plate_text.strip())

        # Draw the contour and text of the number plate detected
        cv2.drawContours(img, [candidate_number_plate], -1, (0, 255, 0),
3)
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
        cv2.putText(img, plate_text.strip(), (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 0), 2)

        cv2.imshow("Number Plate Detection and Recognition", img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
    else:
        print("No number plate detected.")

detect_and_read_plate('car1.jpg')
```