**Vehicle Number Plate Detection Using OCR**

*A PROJECT REPORT*

*Submitted to*

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA**

*In partial fulfillment of requirements for the award of the Degree*

**BACHELOR OF TECHNOLOGY**

*By*

**Adapa Manoj 18HP1A0546**

*UNDER THE ESTEEMED GUIDANCE OF*

**RAKKISU PADMAJA**

## Assistant Professor

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**ANDHRA LOYOLA INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**(AFFILIATED TO JNTU, KAKINADA)**

**VIJAYAWADA**. **JULY-2022**



**ANDHRA LOYOLA INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**VIJAYAWADA**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that the project report entitled **“VEHICLE NUMBER DETECTION USING OCR”** is submitted by **Mr. ADAPA MANOJ** tothe JNTU KAKINADA in partial fulfillment for the award of Degree of Bachelor of Technology in Computer Science and Technology is a bonafide work carried out by them under my supervision during the year 2021-2022.

## (RAKKISU PADMAJA) (Dr. CH. RAJENDRA BABU)

**Assistant Professor Head of the Department**

**Signature of the External Examiner**

**DECLARATION**

We **Mr**. **ADAPA MANOJ**, hereby declare that the project report entitled **“Vehicle Number Plate Detection Using OCR”** is an original work done in the Department of **Computer Science and Engineering**, **Andhra Loyola Institute of Engineering and Technology**, Vijayawada, during the academic year 2021-2022, in partial fulfillment for the award of the Degree of **Bachelor of Technology** in Computer Science and Engineering. We assure that this project is not submitted in any other University or College.

## Roll No Name of the Student Signature

**18HP1A0546 A. Manoj**

**ACKNOWLEDGEMENT**

Firstly, we would like to convey our heartful thanks to Almighty for the blessings on us to carry out this project work without any disruption.

We are extremely thankful to **RAKKISU PADMAJA** our guide throughout Project. We also thank her for the at most independence and freedom of thought given to us during various phases of the project.

We are very much grateful **to Dr. CH. Rajendra Babu**, Associate Professor and H.O.D of C.S.E Department, for his valuable guidance which helped us to bring out this project successfully. His wise approach made us to learn the minute details of the subject. His matured and patient guidance paved away for completing our project with the sense of satisfaction and pleasure.

We are thankful to our Principal **Dr. O. MAHESH** for forte ring an excellent environment in our college and helping us at all points for achieving our task.

We are thankful to our project’s coordinators **RAKKISU PADMAJA and RAZEENA BEGUM SHAIK** for their valuable guidance which helped us to bring this project successfully.

Finally, we would like to convey our heart full thanks to our Technical Staff, for their guidance and support in every step of this project. We convey our sincere thanks to all the faculty and friends who directly or indirectly helped us for the successful completion of this project.

## PROJECT ASSOCIATES

A. Manoj (18HP1A0546)

**ABSTRACT**

Vehicle number plate recognition plays a significant role in many areas. In this paper, an efficient and an amazingly simple method is used to recognize the number plate. In the proposed method, python language is used for image processing using pytesseract. The input image is taken and converted into grayscale image and the processed image is filtered through bilateral filter to remove unwanted characters. In this paper, Canny edge detection method is used to detect the edges of license plate. TESSERACT is used as an Optical Character Recognition (OCR).

**Index Terms**—Tesseract, OCR.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NUMBER** | **FIGURE NAME** | **PAGE NO** |
| **3.2** | **Flow chat of existing system** | **9** |
| **3.3** | **Proposed System** | **11** |
| **5.3** | **Proposed System** | **18** |
| **5.3.1.1** | **Class diagram** | **32** |
| **5.3.1.2** | **Component Diagram** | **33** |
| **5.3.1.3** | **Deployment Diagram** | **34** |
| **5.3.2.1** | **Use case Diagram** | **36** |
| **5.3.2.2** | **Statechart Diagram** | **37** |
| **5.3.2.3** | **Activity Diagram** | **38** |
| **5.3.2.4.1** | **Sequence Diagram** | **40** |
| **5.3.2.4.2** | **Collaboration Diagram** | **41** |
| **7.2** | **Modules Description** | **60** |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | **LIST OF SCREENSHOTS** |  |
| **Screenshot No.** | **Screenshot Name** | **Page No** |
| **9.1.1** | **Interface of uploading image** | **69** |
| **9.1.2** | **Image to be uploaded** | **69** |
| **9.1.3** | **Image uploaded** | **70** |
| **9.1.4**  **9.1.5** | **Printing vehicle number**  **Vehicle number is sent via mail** | **70**  **71** |

|  |  |  |
| --- | --- | --- |
|  | **Table of Contents** |  |
| **Title** | | **Page No.** |
| Acknowledgement | | i |
| Abstract | | ii |
| List of Figures | | iii |
| List of Screenshots | | iv |
| **CHAPTER 1: INTRODUCTION** | | |
| 1.1 | About the project | 2 |
| 1.2 | Purpose | 3 |
| 1.3 | Motivation | 3 |
| 1.4 | Scope | 3 |
| **CHAPTER 2: LITERATURE SURVEY** | | 4-6 |
| **CHAPTER 3: SYSTEM STUDY AND ANALYSIS** 7-11 | | |
| 3.1 | Feasibility Study | 8 |
| 3.2 | Existing System | 9 |
| 3.3 | Proposed System | 10 |
|  |  |  |
| **CHAPTER 4: SYSTEM REQUIREMENTS**   * 1. 4.1 Requirement Specification   2. 4.2 Software Requirements   3. 4.3 Hardware Requirements | | 12-16  13-15  16  16 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER 5: SYSTEM DESIGN** 17-41 | | |
| 5.1. | Introduction | 18 |
| 5.2. | System Architecture | 18 |
| 5.3. | UML Diagrams | 19-34 |
|  | 5.3.1. Structural Diagrams | 31 |
|  | 5.3.1.1. Class Diagram | 31 |
|  | 5.3.1.2. Component Diagram | 32 |
|  | 5.3.1.3. Deployment Diagram | 33-34 |
|  | 5.3.2. Behavior Diagrams | 34-41 |
|  | 5.3.2.1. Use case Diagram | 34-36 |
|  | 5.3.2.2. Statechart Diagram | 36-37 |
|  | 5.3.2.3. Activity Diagram | 37-38 |
|  | 5.3.2.4. Interaction Diagrams | 38-41 |
|  | 5.3.2.4.1. Sequence Diagram | 40 |
|  | 5.3.2.4.2. Collaboration Diagram | 41 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER 6: SOFTWARE ENVIRONMENT** 42-57 | | |
| 6.1. | Introduction | 43 |
| 6.2. | About Python  6.2.1. What is python?  6.2.2. What is python used for?  6.2.3. Data analysis and machine learning  6.2.4. Web development  6.2.5. Automation or scripting  6.2.6. Software testing and prototyping  6.2.7. Pytesseract | 44-57 |

CHAPTER 7: SYSTEM IMPLEMENTATION 58-63

7.1. Introduction 59

7.2. Modules and Description 59-60

7.3. Sample Code 60-36

|  |  |  |
| --- | --- | --- |
| **CHAPTER 8: TESTING** 64-67 | | |
| 8.1 | Introduction | 65 |
| 8.2 | Testing | 65-67 |
|  |  |  |

[CHAPTER 9: EXPERIMENTAL RESULTS 68-71](#_TOC_250002)

[CHAPTER 10: CONCLUSION](#_TOC_250001) 72-73

CHAPTER 11: FUTURE ENHANCEMENTS 74-75

[CHAPTER 12: REFERENCES 76-7](#_TOC_250000)8

**INTRODUCTION**

# 

**CHAPTER-1**

#### 1.1 ABOUT THE PROJECT

With the rocketing Indian population, the number of unlicensed vehicles has witnessed a meteoric rise only aggravating the ever present and persistent traffic problem in the country. This led to a rise in traffic jams and a significant increase in crimes [1]. So, it becomes important to perforce introduce a system to quickly hand out fines. In commercial establishments to permit only authorized vehicles demands substantial number of resources in both financial as well as in terms of time. So, what we need is a robust and efficient system which will capture the image of license plate and will use it to extract information from it. Automatic number plate recognition is a method which is used to extract the characters of license plate numbers from an image[2]. Image processing is used to extract implement this method.

The main steps involved in image processing are:

1. Scaling the input image and if the image is in RGB format it is converted into Grayscale.
2. Image restoration and enhancement.
3. Prepossessing [3] the image which involves many methods like morphological processing, image compression [4].
4. Segmentation procedure which involves the image partition into its constituent parts. [5]
5. Last step involves object detection and recognition. [5]

This project is also based on above steps. Various other methods like canny edge detection and bilateral filter are also used in our project. The image is passed through various methods of image processing and at last tesseract engine is used to read the text obtained from the final image.

**1.2. PURPOSE**

License plate detection is identifying the part of the car that is predicted to be the number plate. Recognition is identifying the values that make up the license plate.

License plate detection and recognition is the technology that uses computer vision to detect and recognize a license plate from an input image of a car.

This technology applies in many areas. On roads, it is used to identify the cars that are breaking the traffic rules. In security, it is used to capture the license plates of the vehicles getting into and out of certain premises. In parking lots, it is used to capture the license plates of the cars being parked. The list of its applications goes on and on.

**1.3. MOTIVATION**

License plate recognition (LPR) plays a significant role throughout this busy world, owing to the rise in vehicles day by day. Stealing of vehicles, breaking traffic rules, coming into restricted space also are increasing linearly, thus to dam this act registration code recognition is intended. Among the fundamental process steps such as detection of number plate, segmentation of characters and recognition of each character, segmentation plays an important art, since the accuracy of recognition is based on how perfect the segmentation is done. To avoid problems like unwanted illumination, tilt that degrades the segmentation which in turn affects the recognition accuracy numerous algorithms are developed for this work. This paper presents a strong technique for localization, segmentation and recognition of the characters within the located plate. Images from still cameras or videos are obtained and regenerated in to grayscale images.

**1.4. SCOPE**

This application detects the number plate of any vehicle and also gives the detected number plate as an output. Apart from this it will also give the text of the number plate.

**LITERATURE SURVEY**

**CHAPTER 2**

Image processing is a moving exploration point for quite a while what is more, has scope for some advancements in innovative applications and for all the significant creating and progressed areas of society like clinical, security, designing, diversion [6]. In India, more than 250 million vehicles are registered and this stat is increasing day by day. Due to increasing number of vehicles the amount of research in this area has been increased [7]. Earlier number plate detection was done by human assistance only which was a tedious task. So now image processing and computer vision is used to detect the number of vehicle plate easily.

Many methods are used to detect number plate. Use of Convolutional Neural Network (CNN) along with classifiers like haarcascade and SVM is increased day by day [8]. The features used in the field of computer vision are Histogram of Oriented Gradients (HOG) and Hough transform [9]. But in this paper, we have used different methods which involves canny edge detection method to detect the edges of the number plate for further processing of image. Distinctive image preprocessing methods are important to accomplish higher exactness utilizing strategies like RGB to dark, obscuring, thresholding and forming is significant. It is additionally utilized for the standardization and balancing the picture.

Text extraction is the main part in license plate detection location so picture preparing turns into the main part that goes before picture to image transformation. The technique for edge detection focuses for text extraction in archive pictures is the executed approach. It has fixed boundaries relegated for a few sorts of pictures like manually written, typewritten, slanted, and so on and it is very quick.

Paper [6] proposes the strategy for recognizing permit plate in a picture taken from fluctuated distance and extraordinary enlightenments utilizing wavelet change and veiling out potential permit part. After the handling is finished at that point the handled picture can be given to the Tesseract OCR Machine to change over the picture into text utilizing order line interface. The main purpose of Tesseract is to detect the text from the image of license plate. Tesseract is limited to different parameters of picture and for this reason image preprocessing is fundamental.

Paper [10] utilizes a powerful strategy for identification of number plates of vehicles by using Automatic License Plate Recognition (ALPR) framework in which amazingly quick vehicle's number plate is additionally distinguished.

Paper [11] uses Convolutional Neural Network (CNN) in which an enormous dataset of arbitrary tags pictures was taken, were prepared, and tried. Paper [12] employments haarcascade classifier and SVM classifier which are very are prepared on highlights caught from picture date close to head district of cruisers. This paper [12] proposes a framework which will help us with identifying whether the motorcyclist is wearing a helmet or not. Paper [13] has a quite different approach for license plate detection. It uses ARM based embedded model along with Raspberry pi [14].

**SYSTEM STUDY AND ANALYSIS**

3, ‘H. G. Zadeh’ [5] proposed further steps, which is image

extraction and segmentation of images for diagnosing cancer cells.

The Gaussian smoothing concept was introduced as a filtering

purpose, previous to applying the ‘Fast Fourier Transform’ (FFT).

Machine Learning. for tumor detection: ‘NN’, ‘Fuzzy. C-mean’

algorithms was introduced for the identification of tumorous cells

[6]. This takes lower computational time but the accuracy also

lowwatersh

**CHAPTER-3**

#### 3.1. FEASIBILITY STUDY

Preliminary investigation examines project feasibility, the likelihood the application will be useful to the user. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging traditional desktop centric applications and porting them to mobile devices. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* + - Operation Feasibility
    - Technical Feasibility
    - Economic Feasibility

* + 1. **Operation Feasibility**

The operational issue usually raised during the feasibility stage of the investigation includes the following:

* **User-friendly**

Client will use the python resources for screens of their various transactions.

* **Security**

Python Production Server uses HTTPS for security. You can configure the security of a server instance to be as broad or specific as required. The instance can simply encrypt the communication channel between it and a client or it can block unauthorized clients from accessing applications.

* **Availability**

This software will be available since it is maintained at one place.

**3.1.2. Technical Feasibility**

The technical issue raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipment’s have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?

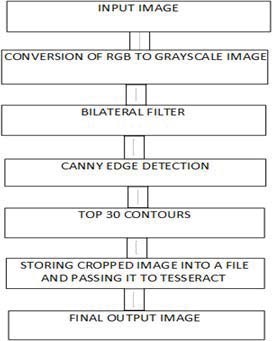
As part of achieving technical competency for developing the proposed system users are required to acquire skill set in python.

**3.1.3. Economic Feasibility**

Usage of Loopback by connecting frontend and backend for communication ensures less cost. Open-source technologies like python usage minimizes the cost for the Developer.

* 1. **EXISTING SYSTEM**

In the existing system, few images of license number plate from dataset were taken for implementation of our code. The car image was taken from a random dataset. The algorithm is divided in five parts as shown in the flowchart:

 **Fig.3.2. Flow chat of existing system**

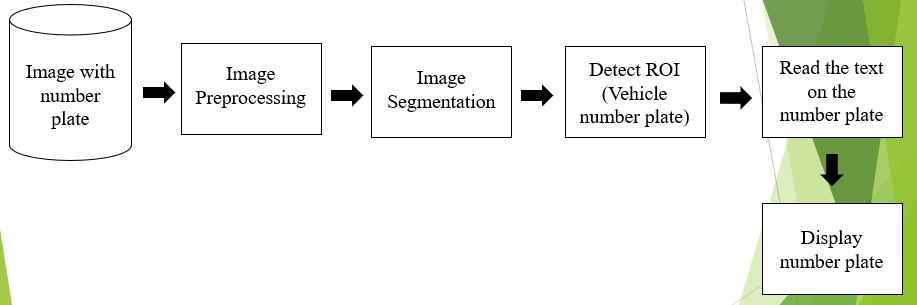
* 1. The above represents the flow chart for the image processing.
  2. The initial step is taking the input image in RGB form from the dataset.
  3. Conversion of the RGB image to gray scale image.
  4. Now the bilateral filter is used to remove background noise while preserving the edges and, we can see some smoothness after applying bilateral filter over the image.
  5. Next canny edge detection process is applied. We have applied this function to detect the edges of license number plate. We have specified the length and breadth of rectangular number plate in our code. The specified edges are 170 and 200.
  6. The next step is detecting all the contours. In this step all contours in the processed image are detected. We can see that front mirror, the two head lights, side mirrors and many more objects of car are considered as contours as shown.
  7. Now among all the detected contours only rectangular con- tours will be detected. After the detection of top 30 contours, these contours will be passed through our conditional function which will detect the area of the number plate. If the contour is in rectangle shape, then only it will be considered as license plate and will go for further process to store that image and cropping license plate number image.
  8. The cropped image of the license number plate.
  9. After passing that image to tesseract we will get the final output of license plate. The final number is displayed on idle terminal. Tesseract will extract string out of image and then it will read that text and will display the final output.

**3.3. PROPOSED SYSTEM**

In proposed system, an image processing model is built to detect the number plate of any vehicle. The process includes four steps. Those are:

* Image processing
* Image Segmentation
* Detect ROI
* Read the number plate

In the initial stage, a car image having vehicle number is uploaded by the user. The uploaded image is resized and converted into a gray scale image for further processing. The preprocessed image is sent to another module i.e., image segmentation in which the region of interest is detected which is none other than the number plate here. The detected number plate is displayed and text is read from the number plate as an output in the console.



**Fig.3.3. Proposed System**

**SYSTEM REQUIREMENTS**

#### CHAPTER-4

## 4.1 REQUIREMENTS SPECIFICATION

## A software requirement specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between the customers and contractors. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development. To develop the software system, we should have clear understanding of software system. To achieve this, we need to continuous communication with customers to gather all requirements.

A good SRS defines the how software system will interact with all internal modules, hardware, communication with other programs and human user interactions with wide range of real-life scenarios. Using the software requirements specification (SRS) document on QA lead, managers create test plan.

**4.1.1. Types**

## There are two types of requirements specification. They are:

## Functional requirements specification

## Non-functional requirements specification

**4.1.1.1. Functional Requirements Specification**

A Functional Requirement (FR) is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements are also called Functional Specification.

## In software engineering and systems engineering, a Functional Requirement can range from the high-level abstract statement of the sender’s necessity to detailed mathematical functional requirement specifications. Functional software requirements help you to capture the intended behavior of the system.

## The present application has been divided in to following modules.

* + - * + Uploading of an image
        + Image Preprocessing
        + Image Segmentation
        + Detect number plate

**4.1.1.2.** **Non- Functional Requirement Specifications**

Non-functional requirement (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of non-functional requirement, “how fast does the website load?” Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

Non-functional requirement allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users is > 10000. Description of non-functional requirements is just as critical as a functional requirement.

* **Performance**

The system must be interactive and the delays involved must be less. So, in every action-response of the system, there are no immediate delays. In case of opening windows forms, of popping error messages and saving the settings or sessions there is delay much below 2 seconds. In case of opening databases, sorting questions and evaluation there are no delays and the operation is performed in less than 2 seconds for opening, sorting, computing, posting > 95% of the ﬁles. Also, when connecting to the server the delay is based editing on the distance of the 2 systems and the Conﬁguration between them so there is high probability that there will be or not a successful connection in less than 20 seconds for sake of good communication.

* **Reliability**

As the system provide the right tools for discussion, problem solving it must be made sure that the system is reliable in its operations and for securing the sensitive details.

* **Safety**

Information transmission should be securely transmitted to server without any changes in information.

* **Security**

The main security concern is for users account hence proper login mechanism should be used to avoid hacking. The tablet id registration is way to spam check for increasing the security. Hence, security is provided from unwanted use of recognition software.

* **Availability**

If the internet service gets disrupted while sending information to the server, the information can be sent again for veriﬁcation.

* **Usability**

As the system is easy to handle and navigates in the most expected way with no delays. In that case the system program reacts accordingly and transverses quickly between its states.

* **Portability**

It is the usability of the same software in different environments. The pre -requirement for portability is the generalized abstraction between the application logic and system interfaces. When software with the same functionality is produced for several computing platforms, portability is the key issue for development cost reduction.

* **Testability**

Software testability is the degree to which a software artifact (i.e., a software system, software module, requirements- or design document) supports testing in a given test context. If the testability of the software artifact is high, then finding faults in the system (if it has any) by means of testing is easier.

**4.2. SOFTWARE REQUIREMENTS**

Operating System : Windows7 & above versions

Coding Language : Python

**4.3.** **HARDWARE REQUIREMENTS**

Processor : i3 & any latest versions.

Hard Disk : 500 GB.

Monitor : 15’’LED

Input Devices : Keyboard, Mouse

RAM : 8GB

**SYSTEM DESIGN**

**CHAPTER-5**

#### 5.1 INTRODUCTION

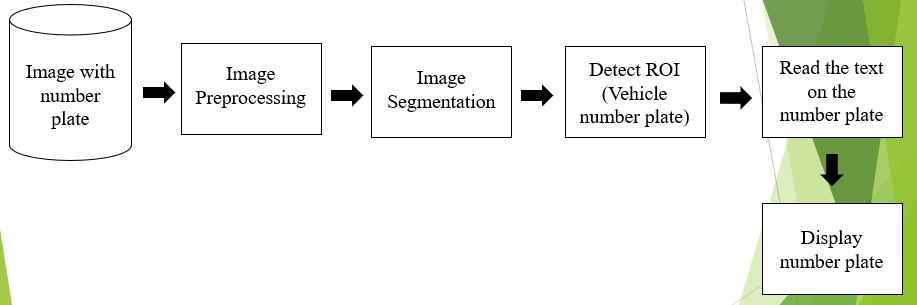
System design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering.

**5.2 SYSTEM ARCHITECTURE**

In proposed system, an image processing model is built to detect the number plate of any vehicle. The process includes four steps. Those are:

* Image processing
* Image Segmentation
* Detect ROI
* Read the number plate

In the initial stage, a car image having vehicle number is uploaded by the user. The uploaded image is resized and converted into a gray scale image for further processing. The preprocessed image is sent to another module i.e., image segmentation in which the region of interest is detected which is none other than the number plate here. The detected number plate is displayed and text is read from the number plate as an output in the console.

**Fig.5.3. Proposed System**

**5.3. UML DIAGRAMS**

**Description**

In the dynamic en-route filtering scheme, the dataflow diagram describes the en-route node receives a report from the source node or the lower associated en-route node and check the integrity of the received report by means of the MAC enclosed in the report. If the validation succeeds then forward the report otherwise drop the report.

**Input Design**

Input design plays a vital role in the life cycle of the software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So, inputs are supposed to be designed effectively so that the errors occurring while feeding as minimized. According to the software engineering concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations.

This system has input screens in almost all the modules. Error messages are developed to alert the user whenever he commits some mistakes and guides him in the right way so that invalid entries are not made. Let us see deeply about this under module design.

Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors. The error is in the input are controlled by the input design. The application has been developed in user-friendly manner. The forms have been designed in such a way during the processing the cursor is placed in the position where must be entered. The user is also provided with in an option to select an appropriate input from various alternatives related to the field in certain cases.

Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can move on to the

subsequent pages after completing all the entries in the current page.

**Output Design**

The output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the client. The output of VPN is the system which allows the project leader to manage his clients in term of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access to each client on the user side depending on the project allotted to him. After completion of a project, a new project may be assigned to the client. User authentication procedures are maintained at the initial stages itself. A new user may be created by the administrator himself or a user can himself register as a new user but the task of assigning projects and validation a new user sets with the administrator only.

The application starts running when it is executed for the first time. The server has to be started. The project will run on the local area network so the server machine will serve as the administrator while the other connected systems can act as the clients. The developed system is highly user friendly and can be easily understood by anyone using it even for the first time.

**Introduction**

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows:

* **User Model View**

1. This view represents the system from the user’s perspective.
2. The analysis representation describes a usage scenario from

the end-user’s perspective.

* **Structural model View**

1. In this model the data and functionality are arrived from inside

the system.

1. This model view models the static structures.

* **Behavioral Model View**
  + 1. It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.
* **Implementation Model View**

i. In this the structural and behavioral as parts of the system are

represented as they are to be built.

* **Environmental Model View**

i. In this the structural and behavioral aspect of the environment in which the system is to be implemented are represented.

As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually. To use UML, it has three basic building blocks that needs to be learnt first before we start drawing the UML diagrams. Those building blocks of UML are –

* Things
* Relationships
* Diagrams

#### THINGS:

Things are the most important building blocks of UML. Things can be:

1. Structural

ii. Behavioral

iii. Grouping

iv. Annotational

#### i. STRUCTURAL THINGS:

The Structural things define the static part of the model. They represent physical and conceptual elements. Following are the brief descriptions of the structural things.

**Class:** Class represents set of objects having similar responsibilities.



**Interface:** Interface defines a set of operations which specify the responsibility of a class.



#### Collaboration: Collaboration defines interaction between elements.

#### Use case: Use case represents a set of actions performed by a system for a specific goal.



#### Component: Component describes physical part of a system.



#### Node: A node can be defined as a physical element that exists at run time.



#### BEHAVIOURAL THINGS:

A behavioral thing consists of the dynamic parts of UML models. Following are the behavioral things:

#### Interaction:

Interaction is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.



#### State machine:

State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change.



#### GROUPING THINGS:

Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available:

#### Package:

 Package is the only one grouping thing available for gathering structural and behavioral things

#### ANNOTATIONAL THINGS:

Annotational things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. Note is the only one Annotational thing available.

#### 

#### Note: A note is used to render comments, constraints etc. of an UML element.



#### RELATIONSHIPS:

In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behavior between the model elements.

#### Dependency-relationship

 Dependency is a relationship between two things in which change in one element also affects the other.

#### Association-relationship

Association is basically a set of links that connects the elements of a UML model. It also describes how many objects are taking part in that relationship.



#### Generalization-relationship

Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes the inheritance relationship in the world of objects



#### Realization-relationship

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility, which is not implemented and the other one implements them. This relationship exists in case of interfaces.



**Types**

UML is linked with object-oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

1. Structural Diagrams
2. Behavioral Diagrams

**5.3.1. STRUCTURAL DIAGRAMS**

They capture static aspects or structure of a system. Structural diagrams include:

1. Class diagram
2. Object diagram
3. Component diagram and
4. Deployment diagram.

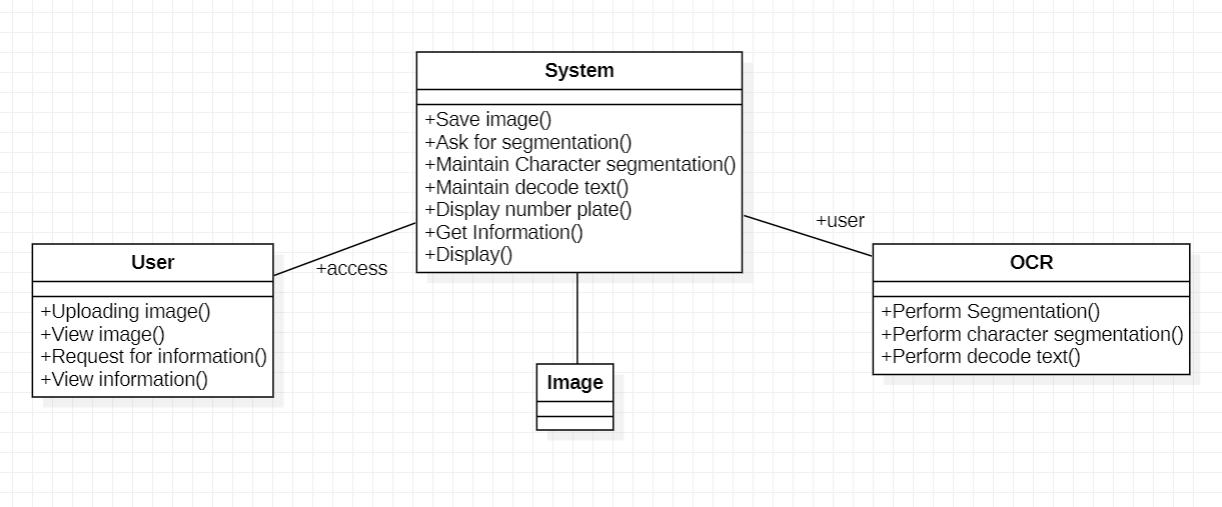
**5.3.1.1. CLASS DIAGRAM**

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

**Purpose of Class Diagrams**

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction. The purpose of the class diagram can be summarized as-

* Analysis and design of the static view of an application.
* Describe responsibilities of a system.
* Base for component and deployment diagrams.
* Forward and reverse engineering.

 **Fig.5.3.1.1. Class Diagram**

**5.3.1.2. COMPONENT DIAGRAM**

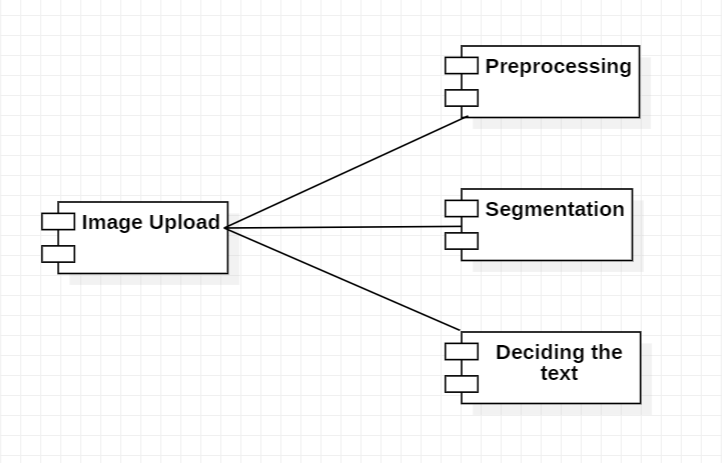
Component diagrams are different in terms of nature and behavior. Component diagrams are used to model the physical aspects of a system. Now the question is, what are these physical aspects? Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node. Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

**Purpose of Component Diagrams**

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as

* Visualize the components of a system.
* Construct executables by using forward and reverse engineering.
* Describe the organization and relationships of the components.



**Fig.5.3.1.2. Component Diagram**

**5.3.1.3. DEPLOYMENT DIAGRAM**

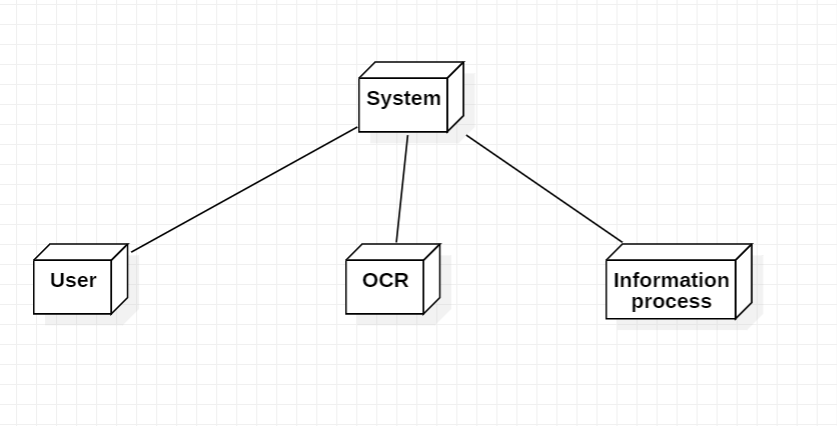
Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

**Purpose of Deployment Diagrams**

The term Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components, where software components are deployed. Most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on the hardware topology of a system. Deployment diagrams are used by the system engineers.

The purpose of deployment diagrams can be described as –

* Visualize the hardware topology of a system.
* Describe the hardware components used to deploy software components.
* Describe the runtime processing nodes.



**Fig.5.3.1.3. Deployment Diagram**

**5.3.2. BEHAVIORAL DIAGRAMS**

They capture dynamic aspects or behavior of a system. Structural diagrams include: Use case diagram, State chart diagram, Activity diagram and Interaction diagram.

**5.3.2.1. USECASE DIAGRAM**

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating. Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior.

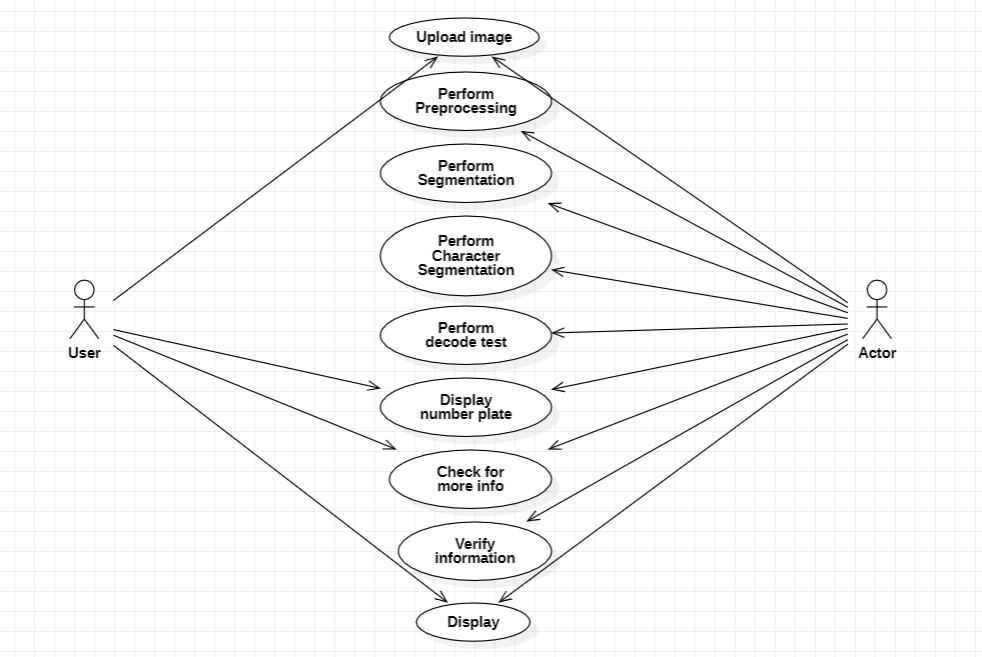
These internal and external agents are known as actors. Use case diagrams consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

**Purpose of Use Case Diagrams**

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State chart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams. When the initial task is complete, use case diagrams are modeled to present the outside view.

In brief, the purposes of use case diagrams can be said to be as follows –

* Used to gather the requirements of a system.
* Used to get an outside view of a system.
* Identify the external and internal factors influencing the system.
* Show the interaction among the requirements are actor.

**Fig.5.3.2.1. Use case Diagram**

**5.3.2.2. STATECHART DIAGRAM**

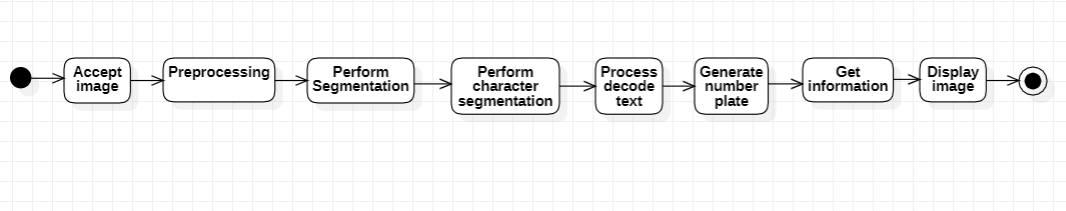
The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system. A Statechart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

**Purpose of Statechart Diagrams**

Statechart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using Statechart diagrams –

* To model the lifetime of a reactive system.
* To describe different states of an object during its life time.
* Define a state machine to model the states of an object.

****

**Fig.5.3.2.2. Statechart Diagram**

**5.3.2.3. ACTIVITY DIAGRAM**

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

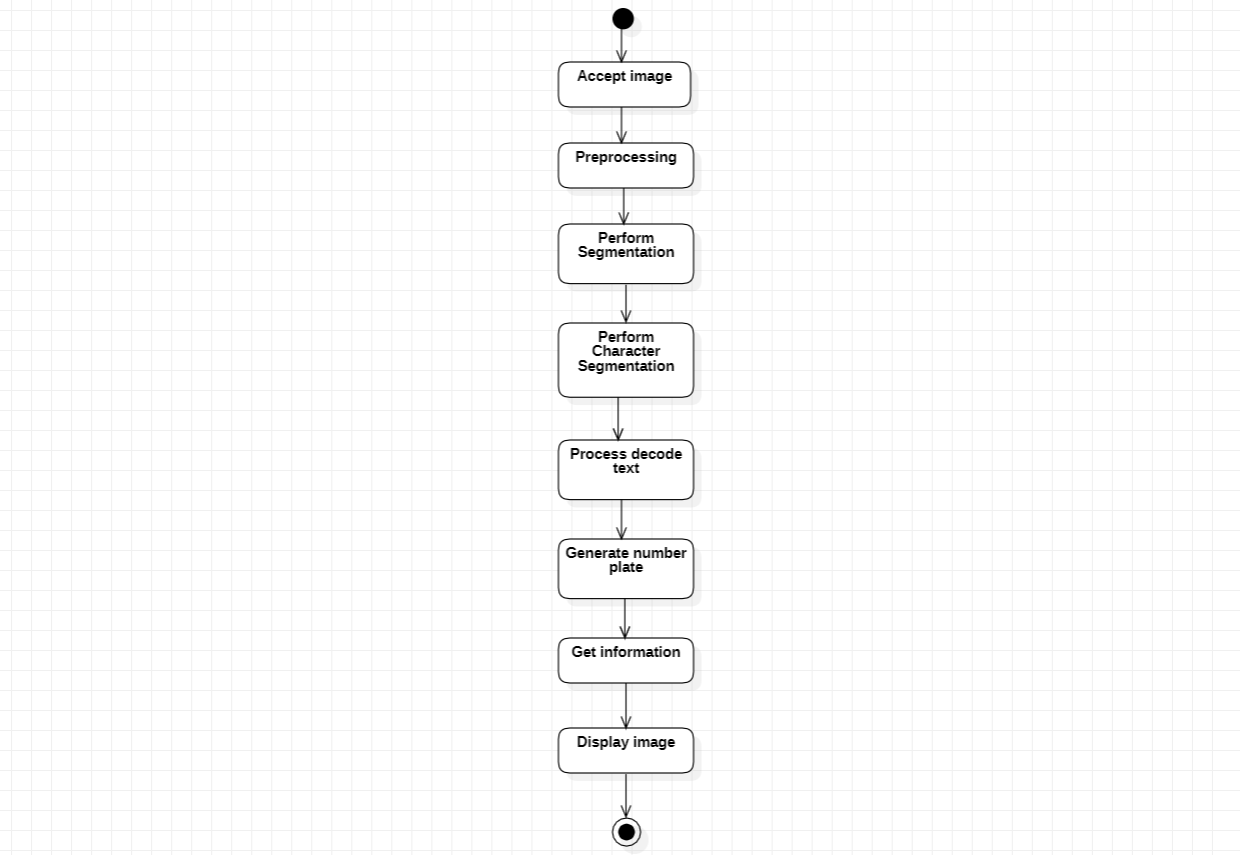
Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

**Purpose of Activity Diagrams**

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

The purpose of an activity diagram can be described as –

* Draw the activity flow of a system.
* Describe the sequence from one activity to another.
* Describe the parallel, branched and concurrent flow of the system.



**Fig.5.3.2.3. Activity Diagram**

**5.3.2.4. INTERACTION DIAGRAMS**

From the term Interaction, it is clear that the diagram is used to describe some type of interactions among the different elements in the model. This interaction is a part of dynamic behavior of the system.

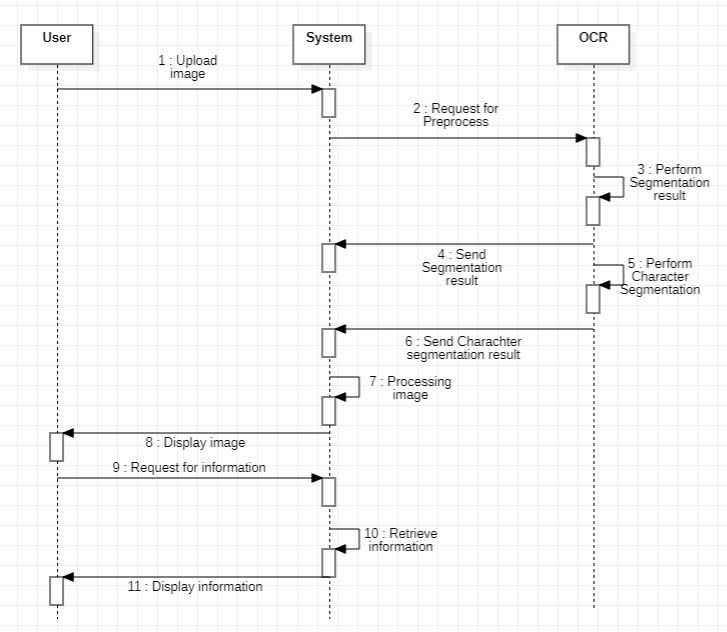
This interactive behavior is represented in UML by two diagrams known as Sequence diagram and Collaboration diagram. The basic purpose of both the diagrams are similar.

**Purpose of Interaction Diagrams**

The purpose of interaction diagrams is to visualize the interactive behavior of the system. Visualizing the interaction is a difficult task. Hence, the solution is to use different types of models to capture the different aspects of the interaction. Sequence and collaboration diagrams are used to capture the dynamic nature but from a different angle.

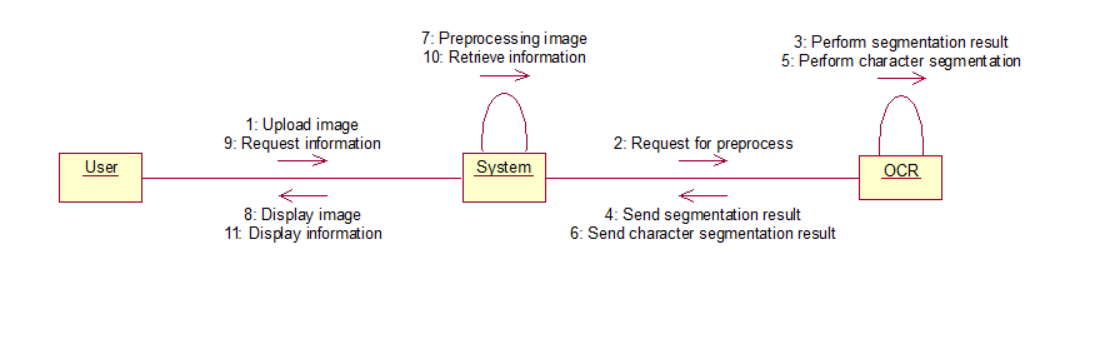
The purpose of interaction diagram is –

* To capture the dynamic behavior of a system.
* To describe the message flow in the system.
* To describe the structural organization of the objects.
* To describe the interaction among objects.

****

**5.3.2.4.1. Sequence Diagram**

**5.3.2.4.2. Collaboration Diagram**

****

**Fig.5.3.2.4.2. Collaboration Diagram**

**SOFTWARE ENVIRONMENT**

**CHAPTER-6**

#### 6.1 INTRODUCTION

Software environment emerged in the middle of the midrange era as a means of improving software quality and productivity through automation. A software environment may be described as an ‘operating system environment and a collection of tools or subroutines. A slightly better definition of software environment is a ‘coordinated collection of software tools organized to support some approach to software development or conform to some software process model’, where software tools are defined as ‘computer programs that assist engineers with the design and development of computer-based systems.

Structured programming environments were created as a means of improving software reliability and productivity using guidelines, code libraries, structured coding, top-down development, chief programmer teams, standards, procedures, documentation, education and metrics. Software factories were soon created to introduce discipline and repeatability, software visualization tools, the capture of customer needs or requirements, automated software testing and software reuse. Computer-assisted software engineering or CASE was also created to enhance software productivity and reliability by automating document production, diagram design, code compilation, software testing, configuration management, management reporting and sharing of data by multiple developers.

#### 6.2. ABOUT PYTHON

Python has become one of the most popular programming languages in the world in recent years. It's used in everything from machine learning to building websites and software testing. It can be used by developers and non-developers alike.

Python, one of the most popular programming languages in the world, has created everything from Netflix’s recommendation algorithm to the software that controls self-driving cars. Python is a general-purpose language, which means it’s designed to be used in a range of applications, including data science, software and web development, automation, and generally getting stuff done.

**6.2.1. What is Python?**

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn’t specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today. A survey conducted by industry analyst firm RedMonk found that it was the second-most popular programming language among developers in 2021.

**6.2.2. What is Python used for?**

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it’s relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

“Writing programs is a very creative and rewarding activity,” says University of Michigan and Coursera instructor Charles R Severance in his book *Python for Everybody.* “You can write programs for many reasons, ranging from making your living to solving a difficult data analysis problem to having fun to helping someone else solve a problem.”

**What can you do with python?**

Some things include:

* Data analysis and machine learning
* Web development
* Automation or scripting
* Software testing and prototyping
* Everyday tasks

### 6.2.3. Data analysis and machine learning

Python has become a staple in data science, allowing [data analysts](https://www.coursera.org/articles/what-does-a-data-analyst-do-a-career-guide) and other professionals to use the language to conduct complex statistical calculations, create data visualizations, build machine learning algorithms, manipulate and analyze data, and complete other data-related tasks.

Python can build a wide range of different data visualizations, like line and bar graphs, pie charts, histograms, and 3D plots. Python also has a number of libraries that enable coders to write programs for data analysis and machine learning more quickly and efficiently, like TensorFlow and Keras.

### 6.2.4. Web development

Python is often used to develop the back end of a website or application—the parts that a user doesn’t see. Python’s role in web development can include sending data to and from servers, processing data and communicating with databases, URL routing, and ensuring security. Python offers several frameworks for web development. Commonly used ones include Django and Flask.

Some web development jobs that use Python includeback-end engineers, full stack engineers, Python developers, software engineers, and DevOps engineers.

### 6.2.5. Automation or scripting

If you find yourself performing a task over and over again, you could work more efficiently by automating it with Python. Writing code used to build these automated processes is called scripting. In the coding world, automation can be used to check for errors across multiple files, convert files, execute simple math, and remove duplicates in data.

Python can even be used by relative beginners to automate simple tasks on the computer—such as renaming files, finding and downloading online content or sending emails or texts at desired intervals.

### 6.2.6. Software testing and prototyping

### In software development, Python can aid in tasks like build control, bug tracking, and testing. With Python, software developers can automate testing for new products or features. Some Python tools used for software testing include Green and Requestium.

## 6.2.7. Tesseract OCR

Tesseract is an open-source text recognition (OCR) Engine, available under the Apache 2.0 license. It can be used directly, or (for programmers) using an API to extract printed text from images. It supports a wide variety of languages. Tesseract doesn't have a built-in GUI, but there are several available from the [3rdParty page](https://github.com/tesseract-ocr/tesseract/wiki/User-Projects-%E2%80%93-3rdParty). Tesseract is compatible with many programming languages and frameworks through wrappers that can be found [here](https://github.com/tesseract-ocr/tesseract/wiki/AddOns). It can be used with the existing layout analysis to recognize text within a large document, or it can be used in conjunction with an external text detector to recognize text from an image of a single text line.

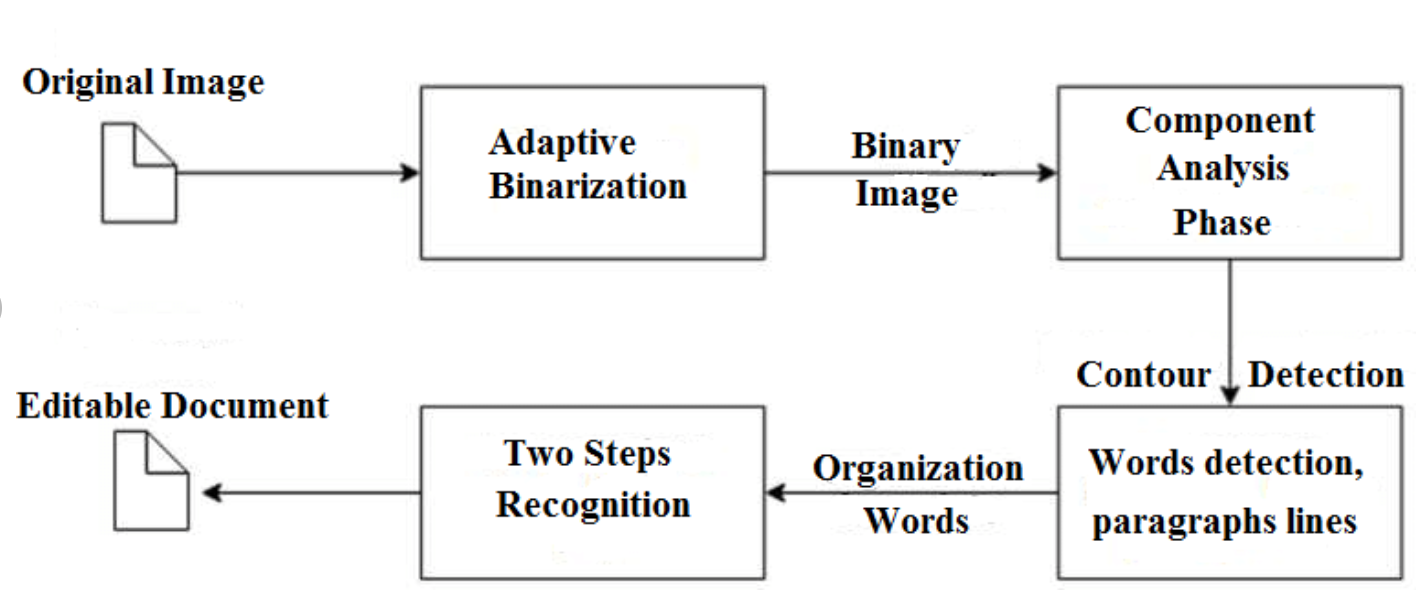
OCR Process Flow to build API with Tesseract from a [blog post](https://medium.com/@balaajip/optical-character-recognition-99aba2dad314)

Tesseract 4.00 includes a new neural network subsystem configured as a text line recognizer. It has its origins in [OCRopus' Python-based LSTM](https://github.com/tmbdev/ocropy) implementation but has been redesigned for Tesseract in C++. The neural network system in Tesseract pre-dates TensorFlow but is compatible with it, as there is a network description language called Variable Graph Specification Language (VGSL), that is also available for TensorFlow.

To recognize an image containing a single character, we typically use a Convolutional Neural Network (CNN). Text of arbitrary length is a sequence of characters, and such problems are solved using RNNs and LSTM is a popular form of RNN. Read this post to learn more about [LSTM](http://colah.github.io/posts/2015-08-Understanding-LSTMs/).

### Technology - How it works

LSTMs are great at learning sequences but slow down a lot when the number of states is too large. There are empirical results that suggest it is better to ask an LSTM to learn a long sequence than a short sequence of many classes. Tesseract developed from OCRopus model in Python which was a fork of a LSMT in C++, called CLSTM. CLSTM is an implementation of the LSTM recurrent neural network model in C++, using the Eigen library for numerical computations.

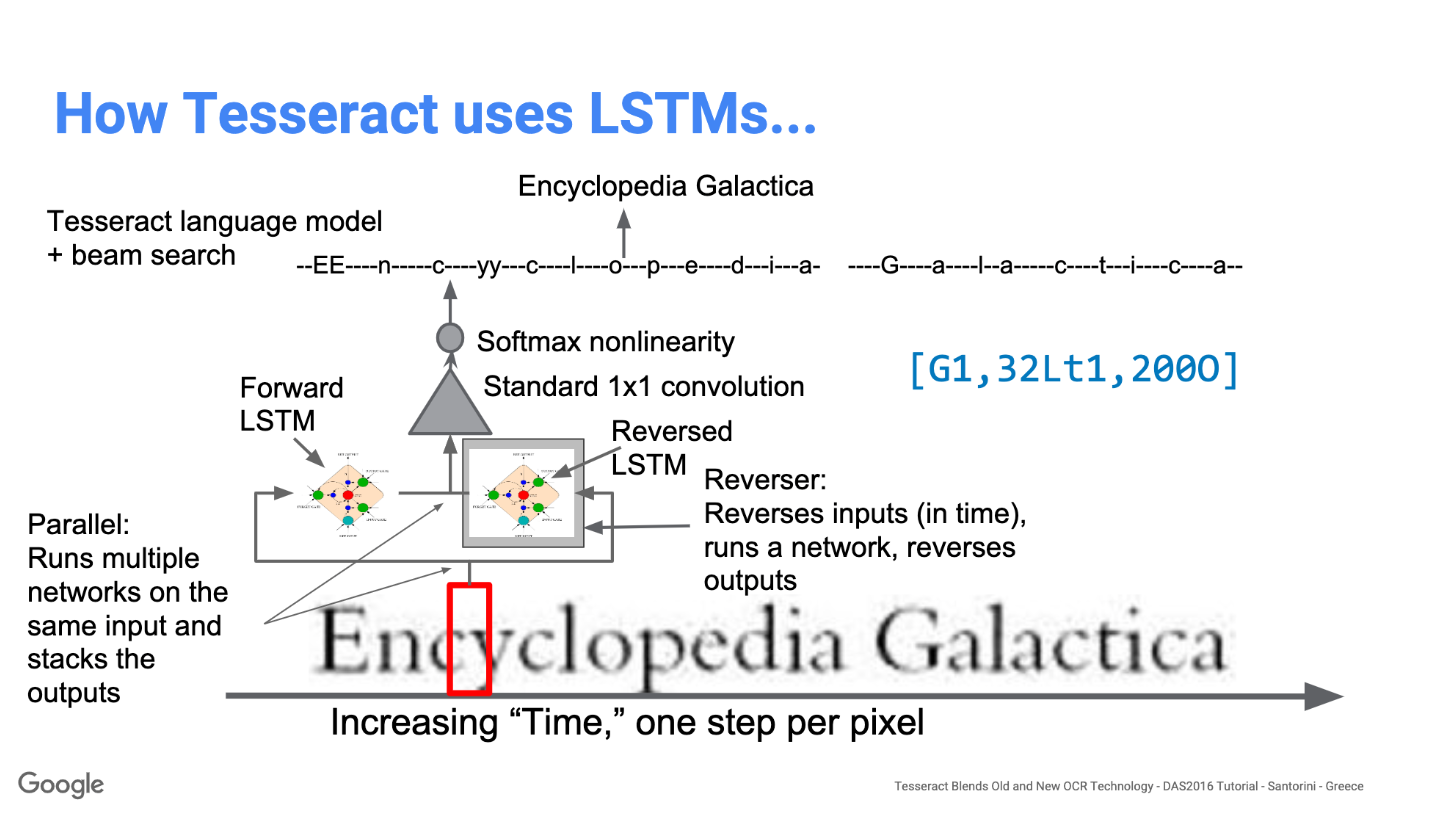
Tesseract 3 OCR process from [paper](https://www.researchgate.net/publication/326016983_VOTING-BASED_OCR_SYSTEM)

Legacy Tesseract 3.x was dependant on the multi-stage process where we can differentiate steps:

* Word finding
* Line finding
* Character classification

Word finding was done by organizing text lines into blobs, and the lines and regions are analyzed for fixed pitch or proportional text. Text lines are broken into words differently according to the kind of character spacing. Recognition then proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text lower down the page.

Modernization of the Tesseract tool was an effort on code cleaning and adding a new LSTM model. The input image is processed in boxes (rectangle) line by line feeding into the LSTM model and giving output. In the image below we can visualize how it works.

How Tesseract uses LSTM model [presentation](https://github.com/tesseract-ocr/docs/blob/master/das_tutorial2016/6ModernizationEfforts.pdf)

After adding a new training tool and training the model with a lot of data and fonts, Tesseract achieves better performance. Still, not good enough to work on handwritten text and weird fonts. It is possible to fine-tune or retrain top layers for experimentation.

### Installing Tesseract

Installing tesseract on Windows is easy with the precompiled binaries found [here](https://digi.bib.uni-mannheim.de/tesseract/). Do not forget to edit “path” environment variable and add tesseract path. For Linux or Mac installation it is installed with [few commands](https://github.com/tesseract-ocr/tesseract/wiki).

After the installation verify that everything is working by typing command in the terminal or cmd:

$ tesseract --version

And you will see the output similar to:

tesseract 4.0.0

leptonica-1.76.0

libjpeg 9c : libpng 1.6.34 : libtiff 4.0.9 : zlib 1.2.8

Found AVX2

Found AVX

Found SSE

You can install the python wrapper for tesseract after this using pip.  
$ pip install pytesseract

Tesseract library is shipped with a handy command-line tool called tesseract. We can use this tool to perform OCR on images and the output is stored in a text file. If we want to integrate Tesseract in our C++ or Python code, we will use Tesseract’s API.

### Running Tesseract with CLI

Call the Tesseract engine on the image with image\_path and convert image to text, written line by line in the command prompt by typing the following:

$ tesseract image\_path stdout

To write the output text in a file:

$ tesseract image\_path text\_result.txt

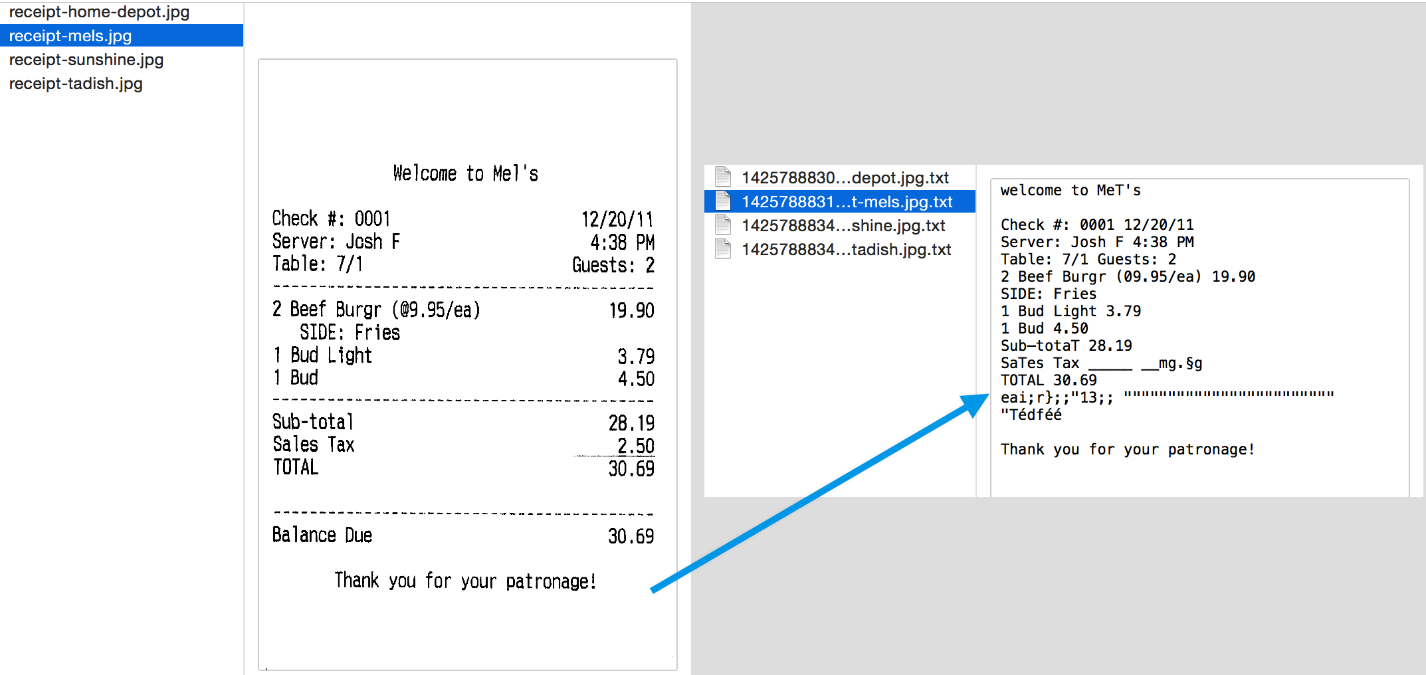
To specify the language model name, write language shortcut after -l flag, by default it takes English language:

$ tesseract image\_path text\_result.txt -l eng

By default, Tesseract expects a page of text when it segments an image. If you're just seeking to OCR a small region, try a different segmentation mode, using the --psm argument. There are 14 modes available which can be found [here](https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality#page-segmentation-method). By default, Tesseract fully automates the page segmentation but does not perform orientation and script detection. To specify the parameter, type the following:

$ tesseract image\_path text\_result.txt -l eng --psm 6

There is also one more important argument, OCR engine mode (oem). Tesseract 4 has two OCR engines — Legacy Tesseract engine and LSTM engine. There are four modes of operation chosen using the --oem option.  
0    Legacy engine only.  
1    Neural nets LSTM engine only.  
2    Legacy + LSTM engines.  
3    Default, based on what is available.

Result of the Tesseract OCR engine

## OCR with Pytesseract and OpenCV

Pytesseract or Python-tesseract is an OCR tool for python that also serves as a wrapper for the Tesseract-OCR Engine. It can read and recognize text in images and is commonly used in python ocr image to text use cases.

It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others.

More info about Python approach read [here](https://github.com/madmaze/pytesseract). The code for this tutorial can be found in this [repository](https://github.com/NanoNets/ocr-with-tesseract).

import cv2

import pytesseract

img = cv2.imread('image.jpg')

# Adding custom options

custom\_config = r'--oem 3 --psm 6'

pytesseract.image\_to\_string(img, config=custom\_config)

### Preprocessing for Tesseract

To avoid all the ways your tesseract output accuracy can drop, you need to make sure the image is appropriately [pre-processed](https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality#image-processing).

This includes rescaling, binarization, noise removal, deskewing, etc.

To preprocess image for OCR, use any of the following python functions or follow the [OpenCV documentation](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_table_of_contents_imgproc/py_table_of_contents_imgproc.html).

import cv2

import numpy as np

img = cv2.imread('image.jpg')

# get grayscale image

def get\_grayscale(image):

return cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# noise removal

def remove\_noise(image):

return cv2.medianBlur(image,5)

#thresholding

def thresholding(image):

return cv2.threshold(image, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)[1]

#dilation

def dilate(image):

kernel = np.ones((5,5),np.uint8)

return cv2.dilate(image, kernel, iterations = 1)

#erosion

def erode(image):

kernel = np.ones((5,5),np.uint8)

return cv2.erode(image, kernel, iterations = 1)

#opening - erosion followed by dilation

def opening(image):

kernel = np.ones((5,5),np.uint8)

return cv2.morphologyEx(image, cv2.MORPH\_OPEN, kernel)

#canny edge detection

def canny(image):

return cv2.Canny(image, 100, 200)

#skew correction

def deskew(image):

coords = np.column\_stack(np.where(image > 0))

angle = cv2.minAreaRect(coords)[-1]

if angle < -45:

angle = -(90 + angle)

else:

angle = -angle

(h, w) = image.shape[:2]

center = (w // 2, h // 2)

M = cv2.getRotationMatrix2D(center, angle, 1.0)

rotated = cv2.warpAffine(image, M, (w, h), flags=cv2.INTER\_CUBIC, borderMode=cv2.BORDER\_REPLICATE)

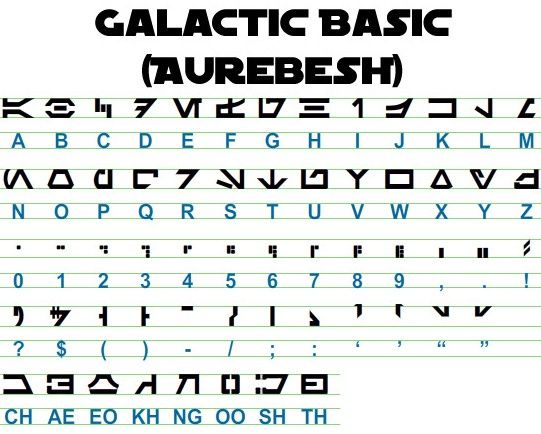
return rotated

#template matching

def match\_template(image, template):

return cv2.matchTemplate(image, template, cv2.TM\_CCOEFF\_NORMED)

Let's work with an example to see things better. This is what our original image looks like -

The Aurebesh writing system

After preprocessing with the following code

image = cv2.imread('aurebesh.jpg')

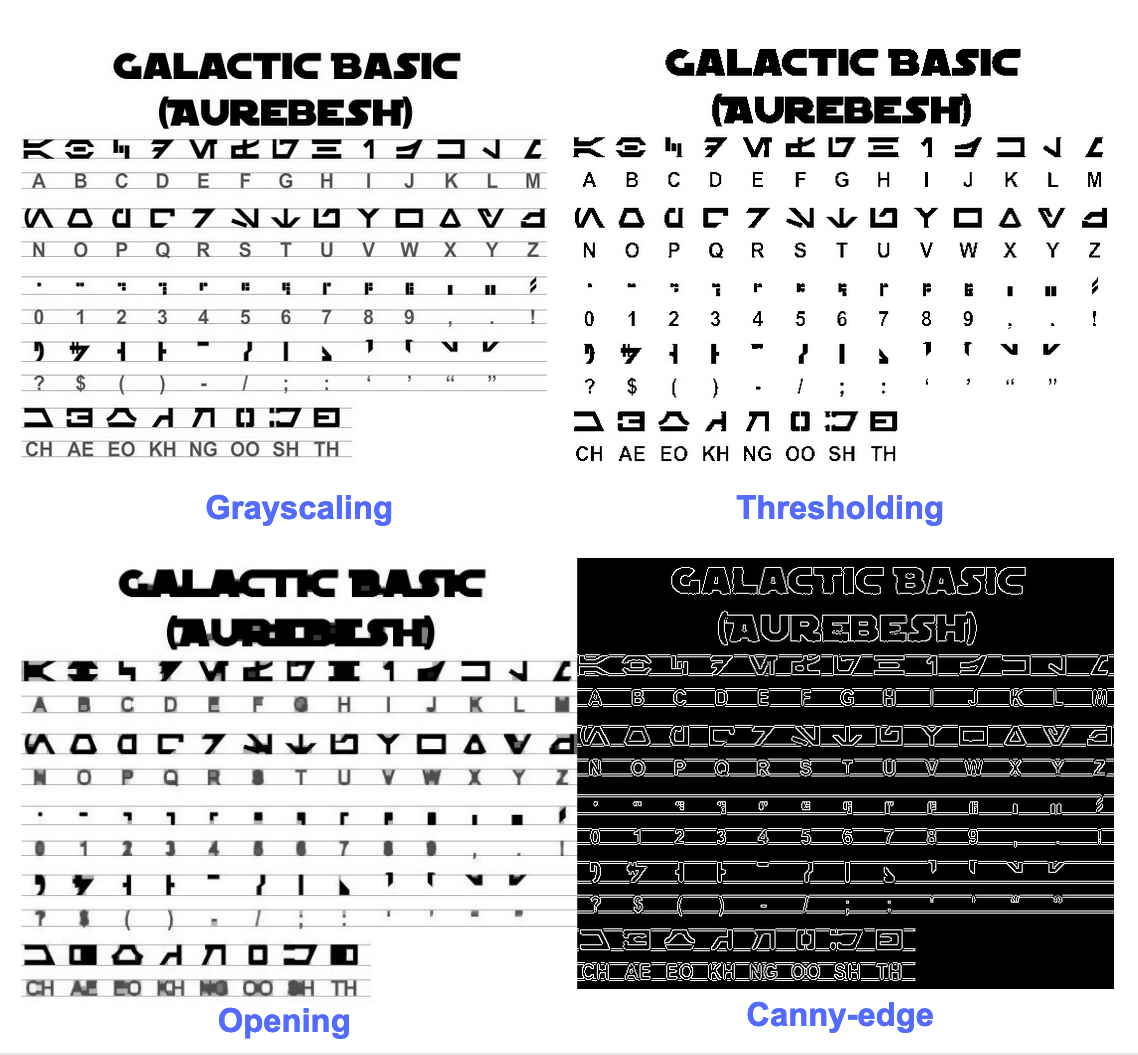
gray = get\_grayscale(image)

thresh = thresholding(gray)

opening = opening(gray)

canny = canny(gray)

and plotting the resulting images, we get the following results.

The image after preprocessing

The output for the original image look like this -

GALACTIC BASIC

(AUREBESH)

RE HFVMEVEIiZwoyv Ze

ABC DE F GH I JK LM

N—0- PQ RST Uv WX

2 | Ff 8 G& Pf fF § 5 op 7

ee

5, jf FF Ty ee ee

=

334 477 OED

Here's what the output for different preprocessed images looks like -

Canny edge image (not so good)-

CAE Cn Cae AS

(AUREBESE)

EA Na

oe SS

(Ne CI (ENE

a, ee oe ea

2

a a A: rc

|, |

a

Sear eo/e

ecm emclomt Cia cuoomct mi im

Thresholded image -

GALACTIC BASIC

(AVREBESH)

RS 7FVMeEVEi1iFf o£

A B C D EF GH IJ K LM

AOoder7Nnvroroava

N O P Q@R S$ TU VW XK Y¥ Z

7 ee For 8 Ro Pf F Boao om #

0 12 3 4 5 6 7 8 9 , . !

>» 1kr7 @ by FEN

2? S$ ( Por Foy of ee

ASGSANDIE

CH AE EO KH NG OO SH TH

Opening image -

GALACTIC BASIC

(AUREZEBELSH)

KEE VTMEUOU EB iw oN es

A BC D EF F @ H | J K LT Ww

AOGdrcrT7WTt HYOAVa4

WO P Q R BS T U VW WK y Z

i J

Oo 1 2 3 46 8 7 SC Ps,

VY ir- -rp,ptUuY?

a a a

AGoOAnNnoOID

CH AE BO KH ®@ OO SH TH

### Getting boxes around text

Using Pytesseract, you can get the bounding box information for your OCR results using the following [code](https://stackoverflow.com/questions/20831612/getting-the-bounding-box-of-the-recognized-words-using-python-tesseract).

The script below will give you bounding box information for each character detected by tesseract during OCR.

import cv2

import pytesseract

img = cv2.imread('image.jpg')

h, w, c = img.shape

boxes = pytesseract.image\_to\_boxes(img)

for b in boxes.splitlines():

b = b.split(' ')

img = cv2.rectangle(img, (int(b[1]), h - int(b[2])), (int(b[3]), h - int(b[4])), (0, 255, 0), 2)

cv2.imshow('img', img)

cv2.waitKey(0)

If you want boxes around words instead of characters, the function image\_to\_data will come in handy. You can use the image\_to\_data function with output type specified with pytesseract Output.

**SYSTEM**

**IMPLEMENTATION**

**CHAPTER-7**

**7.1. INTRODUCTION**

Systems implementation is the process of -

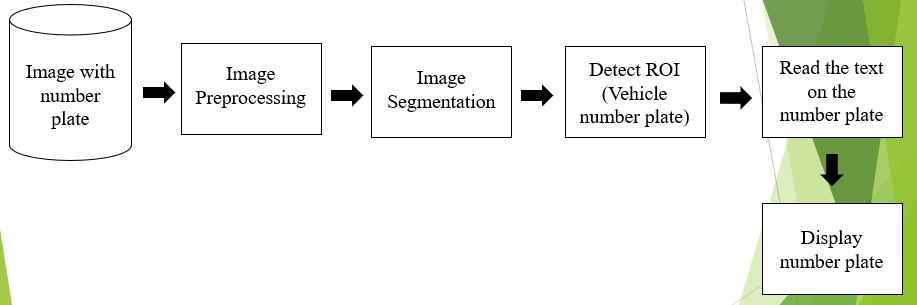
1. defining how the information system should be built (i.e., physical system design),
2. ensuring that the information system is operational and used,
3. ensuring that the information system meets quality standard (i.e., quality assurance).

**7.2. MODULES AND DESCRIPTION**

In proposed system, an image processing model is built to detect the number plate of any vehicle. The process includes four steps. Those are:

* Image processing
* Image Segmentation
* Detect ROI
* Read the number plate

In the initial stage, a car image having vehicle number is uploaded by the user. The uploaded image is resized and converted into a gray scale image for further processing. The preprocessed image is sent to another module i.e., image segmentation in which the region of interest is detected which is none other than the number plate here. The detected number plate is displayed and text is read from the number plate as an output in the console.



**Fig. 7.2. Modules Description**

**7.3. SAMPLE CODE**

**7.3.1 Introduction**

Coding is the process of designing, writing, testing, debugging, and maintaining the source code of computer programs. This source code is written in one or more programming languages. The purpose of programming is to create a set of instructions that computers use to perform specific operations or to exhibit desired behaviors. The process of writing source code often requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal logic.

**7.3.2 Coding**

from PIL.Image import ImageTransformHandler

import cv2

import numpy as np

import pytesseract

import tkinter as tk

import smtplib

from email.message import EmailMessage

from tkinter import filedialog

from tkinter import \*

from PIL import ImageTk, Image

pytesseract.pytesseract.tesseract\_cmd="C:/Program Files /Tesseract-OCR/tesseract.exe"

cascade= cv2.CascadeClassifier("haarcascade\_russian\_plate\_number.xml")

states={"AN":"Andaman and Nicobar",

"AP":"Andhra Pradesh","AR":"Arunachal Pradesh",

"AS":"Assam","BR":"Bihar","CH":"Chandigarh",

"DN":"Dadra and Nagar Haveli","DD":"Daman and Diu",

"DL":"Delhi","GA":"Goa","GJ":"Gujarat",

"HR":"Haryana","HP":"Himachal Pradesh",

"JK":"Jammu and Kashmir","KA":"Karnataka","KL":"Kerala",

"LD":"Lakshadweep","MP":"Madhya Pradesh","MH":"Maharashtra","MN":"Manipur",

"ML":"Meghalaya","MZ":"Mizoram","NL":"Nagaland","OD":"Odissa",

"PY":"Pondicherry","PN":"Punjab","RJ":"Rajasthan","SK":"Sikkim","TN":"TamilNadu",

"TR":"Tripura","UP":"Uttar Pradesh", "WB":"West Bengal","CG":"Chhattisgarh",

"TS":"Telangana","JH":"Jharkhand","UK":"Uttarakhand"}

#initialise GUI

top=tk.Tk()

top.geometry('1920x1096')

top.title('Number Plate Detection')

top.configure(background='#DCDCD')

label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))

sign\_image = Labe(top)

def extract\_num(img\_filename):

global label\_packed

img=cv2.(img\_filename)

img=cv2(img,None,fx=0.8,fy=0.8)

#Img To Gray

gray=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

nplate=cascade.detectMultiScale(gray,1.1,4)

#crop portion

for (x,y,w,h) in nplate:

wT,hT,cT=img.shape

a,b=(int(0.02\*wT),int(0.0\*hT))

plate=img[y+a:y+ha,xb:x+w-b,:]

#make the img more darker to identify LPR

kernel=np.ones((1,1),np.uint8)

plate=cv2.dilate(plate,kernel,iterations=1)

plate=cv2.erode(plate,kernel,iterations=1)

plate\_gray=cv2.cvtColor(plate,cv2.COLOR\_BGR2GRAY)

(thresh,plate)=cv2.threshold(plate\_gray,127,255,cv2.THRESH\_BINARY)

#read the text on the plate

read=pytesseract.image\_to\_string(plate)

read=''.join(e for e in read if e.isalnum())

stat=read[0:2]

#sending mail

msg=EmailMessage()

msg['Subject']='Vehicle Number Detected!!!!'

msg['From']='Message from Connected Device'

msg['To']='projectmsg15@gmail.com'

msg.set\_content(str(read))

server=smtplib.SMTP\_SSL('smtp.gmail.com',465)

server.login("alietprojectbatch15@gmail.com","Project15batch")

server.send\_message(msg)

server.quit()

print(read)

cv2.rectangle(img,(x,y),(x+w,y+h),(51,51,255),2)

cv2.rectangle(img,(x-1,y-40),(x+w+1,y),(51,51,255),-1)

cv2.putText(img,read,(x,y-10),cv2.FONT\_HERSHEY\_SIMPLEX,0.9,(255,255,255),2)

cv2.imshow("plate",plate)

cv2.imwrite("Result.png",img)

cv2.imshow("Result",img)

if cv2.waitKey(0)==113:

exit()

cv2.destroyAllWindows()

def show\_classify\_button(file\_path):

classify\_b=Button(top,text="Detect Image",command=lambda: extract\_num(file\_path),padx=10,pady=5)

classify\_b.configure(background='#364156', foreground='white',font=('arial',10,'bold'))

classify\_b.place(relx=0.79,rely=0.46)

def upload\_image():

try:

file\_path=filedialog.askopenfilename()

uploaded=Image.open(file\_path)

uploaded.thumbnail(((top.winfo\_width()/2.25),(top.winfo\_height()/2.25)))

im=ImageTk.PhotoImage(uploaded)

sign\_image.configure(image=im)

sign\_image.image=im

label.configure(text='')

show\_classify\_button(file\_path)

except:

pass

upload=Button(top,text="Upload an image",command=upload\_image,padx=10,pady=5)

upload.configure(background='#364156', foreground='white',font=('arial',10,'bold'))

upload.pack(side=BOTTOM,pady=50)

sign\_image.pack(side=BOTTOM,expand=True)

label.pack(side=BOTTOM,expand=True)

heading = Label(top, text="Detect Number Plate",pady=20, font=('arial',20,'bold'))

heading.configure(background='#CDCDCD',foreground='#364156')

heading.pack()

top.mainloop()

**TESTING**

**CHAPTER-8**

**8.1. INTRODUCTION**

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully, it will remove all the errors from the software.

**8.2. TESTING**

Testing can also be stated as the process of verifying and validating that a software or application is bug free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently with handling all the exceptional and boundary cases.

**Principles of Testing:**

1. All the test should meet the customer requirements.
2. To make our software testing should be performed by third party.
3. Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
4. All the test to be conducted should be planned before implementing it.
5. It follows pareto rule (80/20 rule) which states that 80% of errors comes from 20% of program components.
6. Start testing with small parts and extend it to large parts.
   * 1. Testing Objectives:

The following are the testing objectives....

* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has a high probability of finding an as yet undiscovered error.
* A successful test is one that uncovers as a yet undiscovered error.

**Steps:**

#### Testing can be divided into two steps. They are –

* **Verification**

It refers to the set of tasks that ensure that software correctly implements a specific function.

* **Validation**

It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

**Verification**: “Are we building the product, right?”

**Validation**: “Are we building the right product?”

**Testing Techniques:**

1) White Box Testing

White box testing is a test case design method that uses the control structure of the procedural design to derive test cases. After performing white box testing it was identified that

* + - * The E-health care system guarantees that all independent paths within the modules have been exercised at least once.
      * It has been exercised all logical decisions on their true and false sides.
    1. 2)Black Box Testing:

Black box tests are designed to uncover errors in functional requirements without regard to the internal workings of a program. Black box testing techniques focus on the information domain of the software.

Black box testing attempts to find errors in the following categories.

* Incorrect or missing functions
* Interface errors
* Errors in the data structures or external database access
* Performance errors

**Types of Testing:**

Testing can be broadly classified into two types:

* **Manual Testing**

Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug.

* **Automation Testing**

Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product.

**Levels of Testing:**

Testing levels can be majorly classified into 4 levels. They are -

* **Unit Testing**

A level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.

* **Integration Testing**

A level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

* **System Testing**

A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements.

* **Acceptance Testing**

A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system’s compliance with the business requirements and assess whether it is acceptable for delivery.

**EXPERIMENTAL RESULTS**

**CHAPTER-9**

**9.1. EXPERIMENTAL RESULTS**



**Fig. 9.1.1 Interface of uploading image**



**Fig. 9.1.2 Image to be uploaded**



**Fig. 9.1.3 Image uploaded**



**Fig. 9.1.4 Printing vehicle number**



**Fig. 9.1.5 Vehicle number is sent via mail**

**CONCLUSION**

**CHAPTER-10**

#### Conclusion:

In proposed system, an image processing model is built to detect the number plate of any vehicle. The process includes four steps. Those are:

* Image processing
* Image Segmentation
* Detect ROI
* Read the number plate

In the initial stage, a car image having vehicle number is uploaded by the user. The uploaded image is resized and converted into a gray scale image for further processing. The preprocessed image is sent to another module i.e., image segmentation in which the region of interest is detected which is none other than the number plate here. The detected number plate is displayed and text is read from the number plate as an output in the console.

**FUTURE ENHANCEMENTS**

**Chapter-11**

**Future Enhancements:**

Further, we will integrate the present application with an external camera which will detect the number plates from the vehicles.

.

**REFERENCES**

**CHAPTER - 12**

**References :**

Here, are some of the references for the project from various sites, journals, papers etc..

1. A. Sasi, S. Sharma and A. N. Cheeran, "Automatic car number plate recognition," 2017 International Conference on Innovations in Informa- tion, Embedded and Communication Systems (ICIIECS), Coimbatore, 2017, pp. 1-6, doi: 10.1109/ICIIECS.2017.8275893..
2. A. Kashyap, B. Suresh, A. Patil, S. Sharma and A. Jaiswal, "Automatic Number Plate Recognition," 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida (UP), India, 2018, pp. 838-843, doi: 10.1109/ICAC- CCN.2018.8748287.
3. Pratiksha Jain, Neha Chopra and Vaishali Gupta, "Automatic License Plate Recognition using OpenCV", International Journal of Computer Applications Technology and Research, vol. 3, no. 12, pp. 756-761, 2014.
4. R. Azad and H. R. Shayegh, "New method for optimization of license plate recognition system with use of edge detection and connected component," ICCKE 2013, Mashhad, 2013, pp. 21-25, doi: 10.1109/IC- CKE.2013.6682800.
5. W. K. I. L. Wanniarachchi, D. U. J. Sonnadara and M. K. Jayananda, "License plate identification based on imagee processing techniques," 2007 International Conference on Industrial and Information Systems, Penadeniya, 2007, pp. 373-378, doi: S. Montazzolli and C. Jung, "Real-Time Brazilian License Plate Detec- tion and Recognition Using Deep Convolutional Neural Networks," 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIB- GRAPI), Niteroi, 2017, pp. 55-62, doi: 10.1109/SIBGRAPI.2017.14.
6. Durgadevi, P., & Vijayalakshmi, S. (2020). Deep Survey and Compar- ative Analysis of Medical Image Processing. Journal of Computational and Theoretical Nanoscience, 17(5), 2321-2329.
7. Kuan Zheng, Yuanxing Zhao, Jing Gu and Qingmao Hu, "License plate detection using Haar-like features and histogram of oriented gradi- ents," 2012 IEEE International Symposium on Industrial Electronics, Hangzhou, 2012, pp. 1502-1505, doi: 10.1109/ISIE.2012.6237313.
8. W. Zhou, H. Li, Y. Lu and Q. Tian, "Principal Visual Word Dis-covery for Automatic License Plate Detection," in IEEE Transactionson Image Processing, vol. 21, no. 9, pp. 4269-4279, Sept. 2012, doi: 10.1109/TIP.2012.2199506.
9. R. R. Palekar, S. U. Parab, D. P. Parikh and V. N. Kamble, "Real time license plate detection using openCV and tesseract," 2017 International Conference on Communication and Signal Processing (ICCSP), Chen- nai, 2017, pp. 2111-2115, doi: 10.1109/ICCSP.2017.8286778.
10. Sarfraz, M.S., Shahzad, A., Elahi, M.A. et al. Real-time automatic license plate recognition for CCTV forensic applications. J Real-Time Image Proc 8, 285–295 (2013). [https://doi.org/10.1007/s11554-011- 0232-7](https://doi.org/10.1007/s11554-011-%200232-7).
11. Y. Kulkarni, S. Bodkhe, A. Kamthe and A. Patil, "Automatic number plate recognition for motorcyclists riding without helmet," 2018 In- ternational Conference on Current Trends towards Converging Tech-

10.1109/ICIINFS.2007.4579205