

PROFESSIONAL TRAINING REPORT

AT

SATHYABAMA INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Technology Degree in Information Technology

By

Vivek Kumar

REG. NO. 40120122



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

**SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY**

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600 119 TAMILNADU

APRIL-2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" Grade by HAAC
Japlaar Nagar, Rajiv Gandhi Salai, Chennai - 600 110.

Phone: 044 - 2450 3180 / 51 / 52 / 54 / 55 Fax: 044 - 2450 2344
www.sathyabama.ac.in



BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Benefice work of **VIVEK KUMAR (Reg. No: 40120122)** who carried out the project entitled —
“**VEHICLE REGISTRATION PLATE DETECTION SYSTEM**” under our supervision from January 2023 to April 2023.

V. Nirmalrani
29/04/2023
Internal Guide

Dr. V. Nirmalrani (M.E., Ph.D.,)

RSM
29/4/23
Head of the Department

Dr. R. Subhashini M.E., Ph.D.,

Submitted for Viva voce Examination held on ... *29-04-2023*

Internal Examiner

External Examiner

DECLARATION

I, **VIVEK KUMAR** hereby declare that the Project Report entitled — **Vehicle registration plate Detection system** done by me under the guidance of **Dr. V. Nirmalrani M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

Date: 29-04-2023

Place: Chennai

Vivek Kumar
Signature of the Candidate

ACKNOWLEDGEMENT

I am pleased to acknowledge our sincere thanks Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA M.E., Ph.D.**, Dean, School of Computing and **Dr. R. SUBHASHINI M.E., Ph.D.**, Head of the Department, Department of Information Technology for providing us the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project **Dr. V. NIRMALRANI M.E., Ph.D.**, for her valuable guidance, suggestion and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-Teaching staff members of the Department of **INFORMATION TECHNOLOGY** who were helpful in many ways for the completion of the project.

ABSTRACT

This project develops an algorithm for automatic vehicle registration plate detection and implemented it using Open-CV library. It is an image processing library which uses number plate to identify the vehicle. The objective is to design an efficient automatic authorized vehicle identification system by using the vehicle number plate. It can be used to implement on the entrance for security control of a highly restricted area like military zones or area around top government offices e.g., Parliament, Supreme Court etc. The algorithm developed here is aimed to be lightweight so that it can be run in real time. With images provided by a USB web camera the system will be able to recognize the number plate under normal condition.

The algorithm is built in three sections; the first is the initial detection of a number plate using edge and intensity features in the image; in the second, the text of the number plate is found; last is the actual character recognition. Optical character recognition technique is used for the character recognition. The resulting data is then used to compare with the records on a database. The major advantages of the system is its real-time capability and that it does not require any additional sensor input (e.g. from infrared sensors) except a video stream. The system is implemented and simulated in MATLAB, and its performance is tested on real image. It is observed that the developed system successfully detects and recognize the vehicle number plate on real images.

CONTENTS

CHAPTER	TITLE	PAGE NO.
	ABSTRACT	
	LIST OF ABBREVIATIONS	
	LIST OF FIGURES	
1	INTRODUCTION	11
1.1	Overview	
1.2	Existing System	
1.2.1	Online ALPR System	
1.2.2	Offline ALPR System	
1.3	Motivation and Technical Relevance	
2	AIM AND SCOPE OF THE PRESENT INVESTIGATION	14
2.1	Aim	
2.2	Scope	
3	LITERATURE SURVEY	15
3.1	Documentation	
3.2	Related works	

3.3	Advantages of the Proposed System	
4	PROPOSED SYSTEMS	18
4.1	Overview of the Proposed System	
4.2	Steps involved in the System with an example	
5	SYSTEM REQUIREMENT SPECIFICATION	22
5.1	Introduction	
5.2	Functional Requirements	
5.3	Non-Functional Requirements	
5.4	Software Requirements	
5.5	Hardware Requirements	
6	DESIGN AND ANALYSIS	25
6.1	System Analysis	
6.2	Module Breakup	
6.3	System Design	
6.4	Use Case Models / Flow Diagrams	
7	IMPLEMENTATION	29
7.1	Methodology	
8	TESTING AND MAINTENANCE	31
8.1	Tests	

8.2	Maintenance	
9	Conclusion	33
9.1	Conclusion	
	APPENDIX	34
	REFERENCES	40

List of Figures

FIG. NO.	FIGURE NAME	PAGE NO.
1.1	Vehicular Growth Graph	12
1.2	Overall View	18
1.3	System Architecture/Ideation Map	19
1.4	Layout	27
1.5	Overall View	28
1.6	Flowchart	28
1.7	Input Data	32

List of Tables

TABLE NO.	TABLE NAME	PAGE NO.
2.1	Module Description	25
2.2	Unit test chart	31

List of Abbreviation

ABBREVIATION	EXPANSION
ANPR	: Automatic Number Plate Recognition
LPR	: License Plate Recognition
OCR	: Optical Character Recognition

CHAPTER - 1

INTRODUCTION

1.1 Overview

Deploying smart cameras for the purpose of video-based traffic surveillance has the advantage of allowing direct on site image processing tasks. The information is extracted from the input data and sent in compressed form to a central node, hence decreasing demands on both communication and computation infrastructure. Furthermore, embedded systems are cheaper than general purpose computers and suitable for deployment in harsh environment. The Automatic Number Plate Recognition (ANPR) was invented at the Police Scientific Development Branch in the UK. There is a need for intelligent traffic management systems in order to cope with the constantly increasing traffic on today's roads. Video based traffic surveillance is an important part of such installations. Information about current situations can be automatically extracted by image processing algorithms. Beside vehicle detection and tracking, identification via number plate recognition is important for a variety of applications. These include, e.g. automatic congestion charge systems, access control, tracing of stolen cars, or identification of dangerous drivers. It is simply the ability to automatically extract and recognition a vehicle number plates characters from an image. In essence it consists of a camera or frame grabber that has the capability to grab an image, find the location of the number in the image and then extract the characters for character recognition tool to translate the pixels into numerically readable character. Furthermore, embedded systems are cheaper than general purpose computers and suitable for deployment in harsh environments due to their physical robustness. Number plate recognition systems in operation today use special hardware like high resolution cameras or infrared sensors to improve the input quality and they operate in controlled settings. A different solution is the continuous analysis and consideration of subsequent frames. The overall system is divided into hardware model and software model. The hardware model consists of a camera to capture the image, a Beagle Board on which the algorithm is executed and for controlling the complete hardware of the system. As the vehicle enters, the camera connected to the Beagle Board through USB port captures the image of the vehicle. The algorithm receives the image and

performs the processing, which yields the vehicle number. This number is then compared to the authorized number to confirm its validity and provides signal to control the system hardware. If the inputted plate contains an authorized number, then the barrier on the entrance will be raised up using motor, green indication light will be switched on and Access Granted will appear on the display, and if the inputted plate contains an unauthorized number then barrier will not be raised, red indication will be switched on and Access denied will appear on the display.

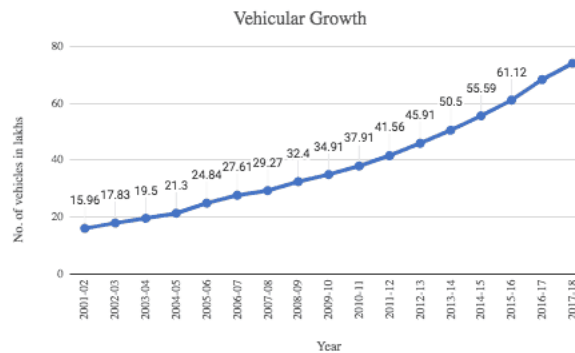


Fig. no: 1.1

EXISTING SYSTEM

Online ALPR System

In an online ALPR framework, the limitation and elucidation of tags occur promptly from the approaching video outlines, enabling Real-time tracking through the surveillance camera.

Example: OpenALPR CloudWatch

Offline ALPR System

A logged off ALPR framework, interestingly, catches the shovel, dumper number plate pictures and stores them in a concentrated information server for further preparation, i.e., for translation of vehicle number plate.

Example: OpenALPR Library

1.2. Motivation and Technical Relevance

The major advantages of our system over all others are its real-time capabilities in city scenarios and its ability to operate under daytime conditions with sufficient daylight or artificial light from street lamps. The usage of active infrared light is popular because the light is reflected by the license plates only. By using a camera and special filters, the detection of the plates and subsequent character segmentation is relatively easy. However, the usage of alternative light sources comes at additional costs. Thus, one prerequisite during system design was to build a system which operates with conventional off-the shelf video cameras and without additional lighting, in favor of the feasibility to deploy and setup the system within a short time. The major reason why we have focused on cars driving at residential speeds is the lack of a digital camera with a high-speed shutter. The drawback of using standard TV cameras is that motion blur is a critical issue, so that reasonable character recognition becomes almost impossible when vehicles are moving at high speeds. Note that this is rather a restriction of the image acquisition device than a deficit of the system. Our system runs fully autonomous and embedded on a smart camera, and thus makes integration into an existing system, e.g., for access control of parking garages, possible.

CHAPTER – 2

AIM AND SCOPE

AIM

The main purpose behind this is uses optical character recognition on images of vehicle registration plates to read the vehicle's registration number. An automatic license plate recognition system applies different image processing techniques to quickly and automatically identify vehicles in video or photo footage.

Scope

This project is developed for detecting license plate from the vehicle and to store the extracted characters of the number plate in database along with their timestamp. The scope of this project is to develop real time number plate recognition system which can be implemented in vehicle tracking, traffic monitoring, automatic payment of tolls on highways or bridges, surveillance systems, tolls collection points, and parking management systems.

CHAPTER - 3

LITERATURE AND SURVEY

2.1. Documentation

Generally, license plate recognition consists of two separate parts. First, plates must be located within the image, termed license plate detection followed by license plate character recognition, determining the plate number.

2.1.1. Related Works

Different methods for the detection of license plates can be applied. Shapiro et al. use a mixture of edge detection and vertical pixel projection for their detection module. In the work of Jia et al. color images were segmented by the MeanShift algorithm into candidate regions and subsequently classified as plate or not. The AdaBoost algorithm was used by Dlagnekov and Belongie for license plate detection on rather low-resolution video frames. Similar features like those introduced in were added to the classical set of Haar features, and the located plate was tracked over a sequence of frames. Matas and Zimmermann [13] proposed a different approach for the localization of license plates. Instead of using properties of the plate directly, the algorithm tries to find all character-like regions in the image. This is achieved by using a region-based approach. Regions are enumerated and classified due to a region descriptor using a neural network. If a linear is a combination of character-like regions is found, the presence of a whole license plate is assumed. The method described above can be applied to the segmentation- task of character recognition as well. Shapiro et al. use adaptive iterative thresholding and analysis of connected components for segmentation. The classification task is performed with two sets of templates. Rahman et al. used horizontal and vertical intensity projection for segmentation and template matching for classification. Dlagnekov and Belongie [6] use the normalized cross correlation for classification by searching the whole plate, hence skipping segmentation.

Those systems described above have not been specifically designed for embedded systems. A lot of commercial mobile systems exist, most of them equipped with special hardware to improve the quality of the input image. Kamat and Ganesan implemented a license plate detection system on a DSP using the Hough transform. Kang et al. implemented a vehicle tracking and license plate recognition system on a PDA. A FPGA was used by Bellas et al. to speed-up parts of their license plate recognition system. A comparison of different LP detection and recognition systems is difficult as each one is subject to differing prerequisites. Furthermore, the lack of a common evaluation database makes evaluations and comparisons unfair. Our focus was on the architecture and implementation of a complete LPR system on our embedded platform. Thus, we have omitted a detailed discussion of algorithm complexity, power efficiency and accuracy. We have only cited those systems which are somehow related to mobile devices or embedded architectures, or which have recently introduced new techniques to the area of LPR.

2.1.2. Advantages of the proposed system

- Easy to understand and use
- Platform independent
- Faster Version
- Light size
- Portable

Algorithm 1 Optical Character Recognition Algorithm

- 1: Create an image model library that contains all the letters A-Z and numbers 0-9.
- 2: Create templates a: Read the letters and numbers (characters) from the library b: Break the image matrix of characters up into a cell array of matrices 42x 24 c: Save as templates
- 3: Read the image (only the number plate portion)

- 4: Convert to gray scale
- 5: Convert the gray image to binary format
- 6: Remove all objects containing fewer than 30 pixels
- 7: Open a text file to write
- 8: Compute the number of letters in the template file a: separate lines in the text (column and row segmentation) b: extract letter c: resize the letter (same size of template) d: computes the correlation between templates and the letter e: convert image to text f: concatenate the letters one by one 9: Open the text file to read

Algorithm 1: Detection Algorithm

- 1: Read image
- 2: Enhance the yellow region to white and others to black? convert image from rgb to hsv? if Hue! 0.2, Saturation? 0.4 and Value ?0.6 then it is likely that that region is yellow? so make this region to white and all others to black
- 3: Process this black and white image for large areas with white color ? perform morphological filling to remove small hole like regions ? search for large connected regions with minimum 20 x 20 pixels (this can be changed) ? separate such regions and store in a structure ? Individually process these regions ?? find out the locations in the region with alternate whites and blacks. This can be found by counting number of white pixels in each column ?? identify the points with maximum white pixels, these are the separation between the characters ?? take the regions between two such points as one character. All such characters are stored to a structure.
- 4: Repeat the above steps for white color also? after converting to hsv format if Hue! 0.2 saturation!0.1 and value?0.8 is considered as white color.

CHAPTER - 4

PROPOSED SYSTEM

3.1 Overview of the proposed system

Figure below shows the block diagram of the system. The camera captures the image and gives the image to the Beagle Board. Beagle Board is the main part of the system on which the algorithm is executed. For executing the algorithm, the operating system is loaded into the beagle board. Then the camera, keyboard, mouse and display is interfaced into the system. The algorithm is first developed in the Matlab and simulated. Then convert that algorithm to run on the Beagle board. The algorithm takes the image from the camera, cut the number plate portion from the image (detection), and from the number plate characters are recognized. Then the characters are compared with the numbers in the database (authorized numbers). If the number is present on the database Access Granted is displayed on the DVI monitor. Otherwise, Access denied will display. There is also an option to add and delete the numbers from the database whenever needed.

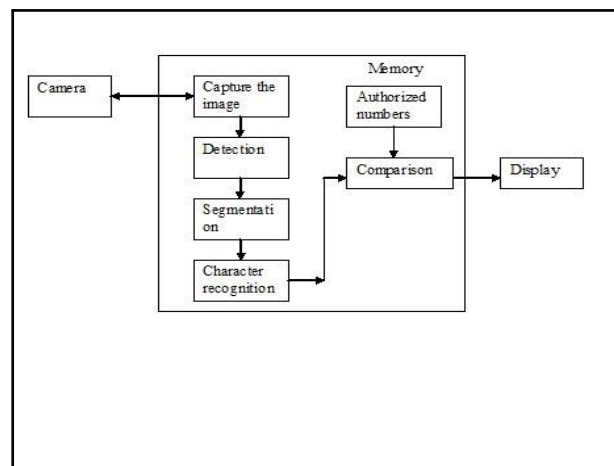


Fig. 1.2 : Overall View

System Architecture/Ideation Map

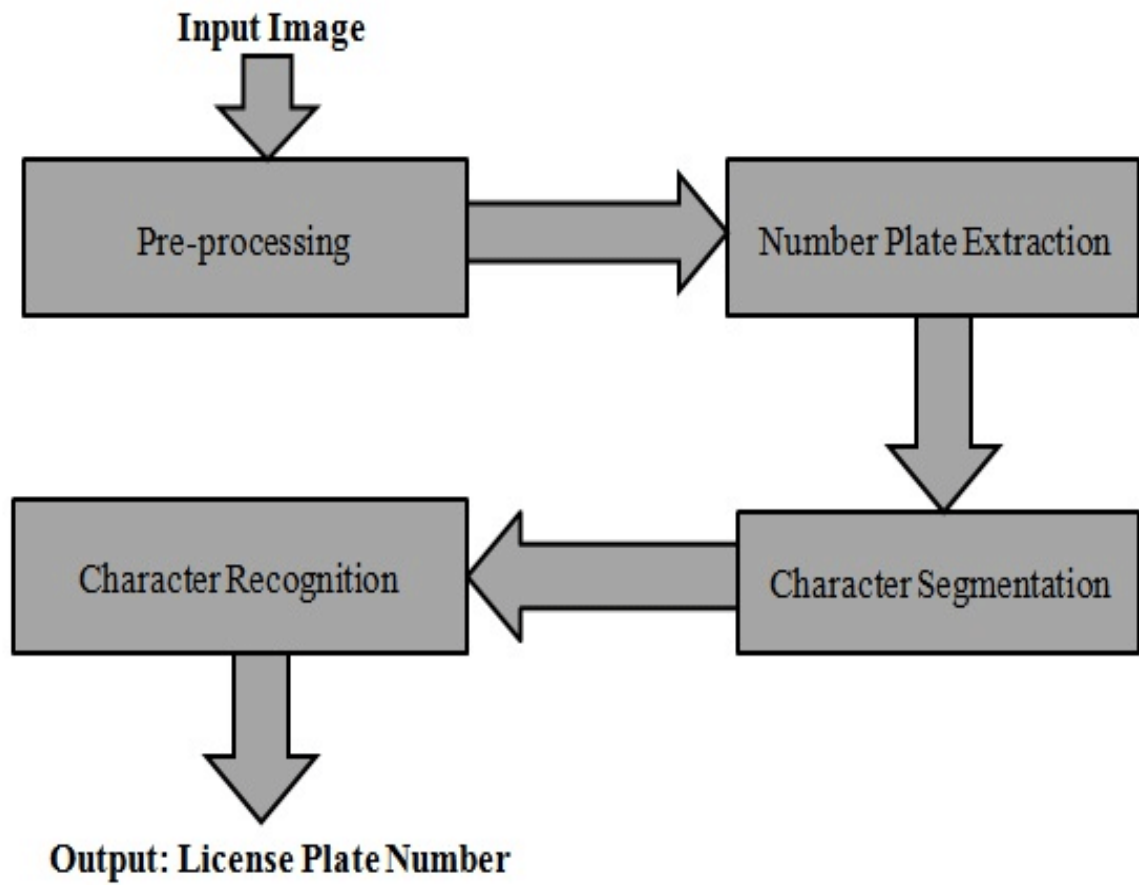


Fig. no: 1.3 Architecture of Process

3.2 Steps involved in the system with an example



Figure 3 Images taken using USB cameras



Figure 4 ROI detection using yellow search algorithm



Figure 5 Vehicle number plate extraction using masking algorithm

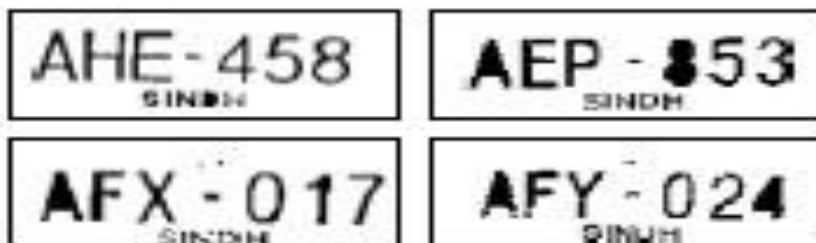


Figure 6 Binary image



Figure 7 Inverted binary image



Figure 8 Line separation using row segmentation

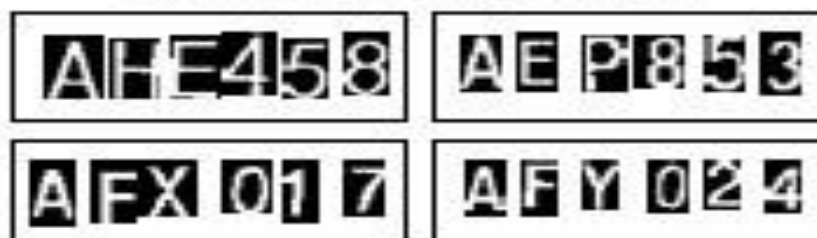


Figure 9 Character separation using column segmentation

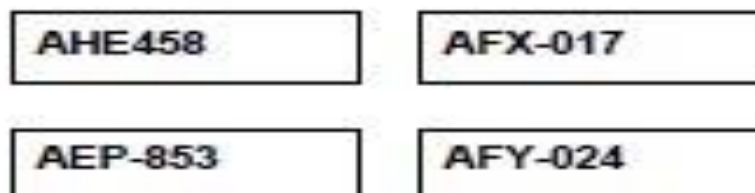


Figure 10 Recognize character using OCR

CHAPTER - 5

System Requirements Specification

5.1 INTRODUCTION

It specifies the hardware and software requirements that are required in order to run the application properly. The Software Requirement Specification (SRS) is explained in detail, which includes overview of dissertation as well as the functional and non-functional requirement of this dissertation.

A SRS document describes all data, functional and behavioral requirements of the software under production or development. SRS is a fundamental document, which forms the foundation of the software development process. It is the complete description of the behavior of a system to be developed. Requirement Analysis discusses the conditions to be met for a new or altered product.

Requirement Analysis is critical to the success to a development project. Requirement must be documented, measurable, testable, related to in identified business needs or opportunities, and defined to a level of detail sufficient for system design.

The SRS functions as a blueprint for completing a project. The goal of preparing the SRS document is to:

- Facilitate communication between the customer, analyst, system developers, maintainers.
- To form a foundation for the design phase.
- Support system testing facilities.
- Controlling the evolution of the system.

5.2 Functional Requirements

1. Capabilities for automatic recognition of vehicles, number plates localization and feature extraction
2. Store the features extracted from the license plate

3. Retrieve, modify information if there is a need for evidence in disputes or crime situations

5.3 Non-Functional Requirements

1. Secure access of confidential data (user's details)
2. 24 X 7 availability and should be efficient
3. Better component design to get better performance at peak time.
4. Flexible service-based architecture will be highly desirable for future extension.
5. The system must display necessary information in case of failure preventing system breakdown.

5.4 Software Requirements

1. Python 3.7 & Open CV

5.5 Hardware Requirements

1. PC/ Laptop/Server
2. Surveillance Camera

Python

Python is a high-level programming language that is widely used for web development, data analysis, artificial intelligence, and scientific computing. Python is an interpreted language, meaning that the code is executed line by line and does not need to be compiled beforehand. It is a dynamically typed language, which means that variables are not declared with a specific data type, and can change their type during runtime. Python's syntax is clean and concise, using indentation to indicate block structure instead of curly braces or keywords. It has a large and active community of developers, who have contributed to an extensive library of modules and packages that can be easily installed with pip. Python supports object-oriented, functional, and procedural programming paradigms, making it a versatile language.

It also has support for multiple programming paradigms such as Aspect-oriented, Meta programming and more. Python has a large and diverse ecosystem of frameworks and tools, such as Django and Flask for web development, NumPy and Pandas for data analysis, and TensorFlow and PyTorch for machine learning and artificial intelligence.

Open-Source Computer Vision Library (OpenCV)

The OpenCV Library is mainly aimed at real time computer vision. Some example areas would be Human-Computer Interaction (HCI); Object Identification, Segmentation, and Recognition; Face Recognition; Gesture Recognition; Motion Tracking, Ego Motion, and Motion Understanding; Structure From Motion (SFM); and Mobile Robotics.

Features:

Image data manipulation (allocation, release, copying, setting, conversion). Image and video I/O (file and camera-based input, image/video file output). Matrix and vector manipulation and linear algebra routines (products, solvers, eigenvalues, SVD). Various dynamic data structures (lists, queues, sets, trees, graphs). Basic image processing (filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids). Structural analysis (connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation). Camera calibration (finding and tracking calibration patterns, calibration, fundamental matrix estimation, homography estimation, stereo correspondence). Motion analysis (optical flow, motion segmentation, tracking). Object recognition (eigen-methods, HMM). Basic GUI (display image/video, keyboard and mouse handling, scroll-bars). Image labeling (line, conic, polygon, text drawing)

CHAPTER - 6

Design and Analysis

6.1 System Analysis

The system consists of two modules: License Plate Detection and License Plate Character Recognition.

6.2 Module Breakup

Module	Description
Module1: License plate Detection.	detecting, tracker and post processing
Module2: License plate Character Recognition.	segmentation, classification, post processing

Table No. 2.1: Module Description

The detection module consists of three parts: the detecting, the tracking and the postprocessing step. Detector: The license plate detector is based upon the framework proposed by Viola and Jones . Viola and Jones created a fast object detection system by combining the AdaBoost-algorithm with Haar-like features, a classifier cascade and the integral image for fast feature computation. Unlike the original framework, the RealBoost is used for this detector, which provides confidence rated predictions instead of binary class-labels. Furthermore, the classifier cascade is improved by using inter stage feature propagation as proposed by Sochman and Matas. The feature-pool is extended with features based on edge orientation histograms as proposed by Levi and Weiss. Post-processing: The exhaustive search technique of the Viola and Jones detector leads to multiple detections for a single license plate. Therefore, post-processing methods have to be applied in order to merge those detections. For this purpose, the rather simple

non-maximum suppression is used. The non-maximum suppression merges multiple detections, considering the confidence value of each detection. In our case all overlapping detections are substituted by the single detection with the maximum confidence value. If the false positive rate is not too high, this method achieves sufficient results. Nonetheless, the final detection does not solely contain the license plate but parts of the surroundings as well. Tracker: A tracker is integrated into the system in order to limit the search of the detector to certain areas of the input image. This requires a rough calibration of the scene. Figure 2 illustrates a possible setting. Objects are assumed to enter from the top. A static region of interest is defined where the Viola and Jones detector search is performed on each frame. For each new detection a tracker is initialized, which predicts the new objects position on the next frame. This allows for scanning solely within a small area around the estimated position. The update and prediction operations of the tracker require less computation time than a detector scan for the whole image. Therefore, the integration is indeed useful. Furthermore, the tracking information will be important.

License Plate Character Recognition Module: License plate character recognition consists of three separate parts. First, characters are isolated with a region-based approach, followed by character classification using a support vector machine (SVM). Finally, during postprocessing, a priori knowledge about the expected license plates and information provided by the tracker is used to improve the classification result. Segmentation: A region-based approach is used for character segmentation. Due to dark characters on white background, region-growing seeded with lowest intensity values is applied. In order to determine which regions contain a character two approaches are used. Similar to [13] a region descriptor is computed incrementally, consisting of compactness, entropy of the grayscale histogram, and central invariant statistical moments. Descriptors are classified using support vector classification. knowledge about the expected license plates and information provided by the tracker is used to improve the classification result. Classification: Support vector classification is used for character classification.

Segmented regions are scaled to a common patch size. The feature vector consists of direct pixel values. Since character recognition is a multi-class problem a combination of binary classifiers is required. Post-Processing: In this step a priori knowledge about the structure of license plates is exploited i.e., spacing between

subsequent characters and furthermore the syntax of the final character string. Needless to mention that this applies only in cases where the nationality of the license plate is known in advance or can be derived beforehand. Additionally, classification results of subsequent frames are combined utilizing the history provided by the tracking module. A simple majority voting individually for each position within the character string is applied. The character achieving the highest count is selected.

6.3 System Design

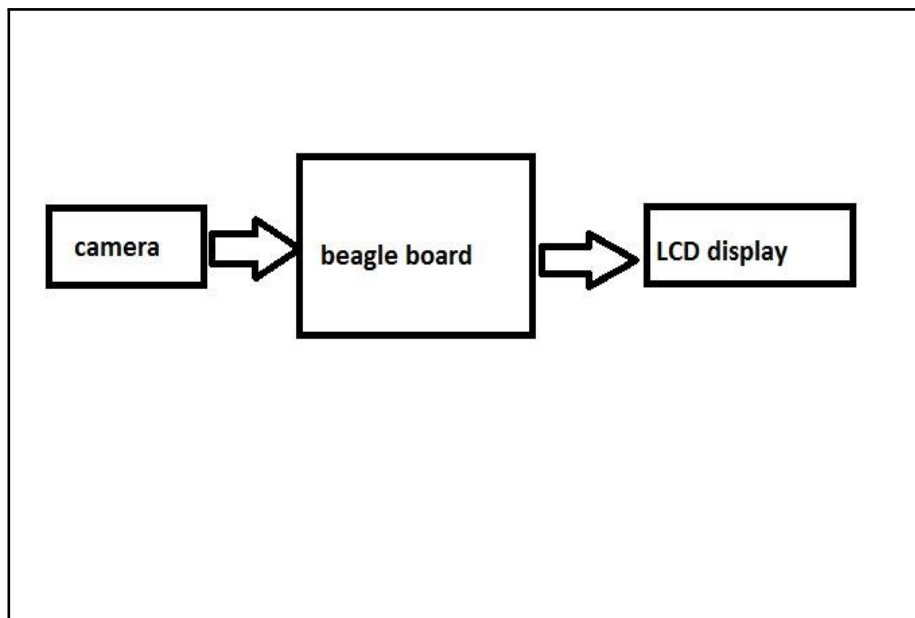


Fig. No. 1.4: Layout

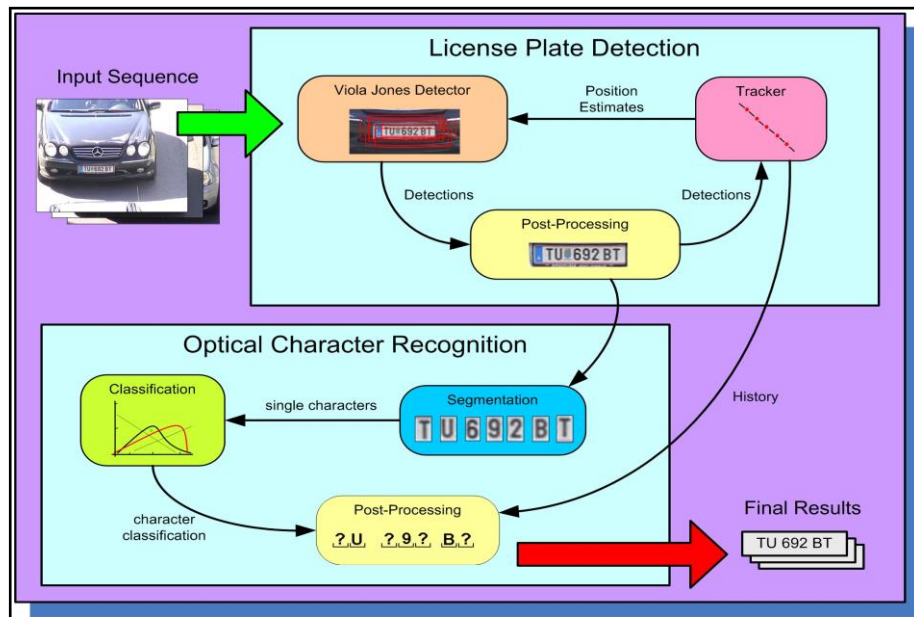


Fig 1.5 : Overall View

6.4 Use Case Models / Flow Diagrams

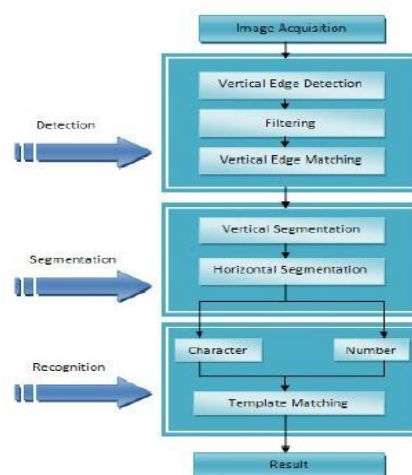


Fig. 1.6 : Flowchart

CHAPTER - 7

IMPLEMENTATION

7.1 PROCESS UNDER

- Loading and data pre-processing
- In this we train the model on the training data and validate it on the validation data. For this we need to create a validation set. This is done by creating a partition in the training set data
- Defining Model architecture
- We must define and elaborate how our model looks by answering questions like
 - How many convolutional layers are needed?
 - What should be the function for activation applied in each layer?
 - How many hidden units do we want in each layer?
- Training the model
- To train a model, we need to
 - Training the images and even map their corresponding true labels with validation images mapped with their corresponding true labels.
- Estimation of performance
- Finally, we load the test data and go through the pre-processing step
- We at that point foresee the classes for these pictures utilizing the prepared model
- Image Acquisition
- Image Desaturation
- Image Thresholding
- Gaussian Filter
- Morphological Transformation
- Segmentation
- Character Recognition

METHOLODOGY

- In this project we used Python, Library- OpenCV, Pytesseract.
- One of the reasons for choosing OpenCV in this project is that it OpenCV is an image processing library of python which enables the detection of desired objects in an image/video.

CHAPTER – 8

TESTING AND MAINTENANCE

7.1 Tests

No	Unit Name	Test Status
1	Detection module	Complete
2	Recognition Module	Partial

Table No. 2.2: Unit Test Chart

7.2 Maintenance

The implementation works quite well however, there is still room for improvement. The camera used in this project is sensitive to vibration and fast changing targets due to the long shutter time. The system robustness and speed can be increase if high resolution camera is used. The OCR methods used in this project for the recognition is sensitive to misalignment and to different sizes, the affine transformation can be used to improve the OCR recognition from different size and angles. The statistical analysis can also be used to define the probability of detection and recognition of the vehicle number plate.

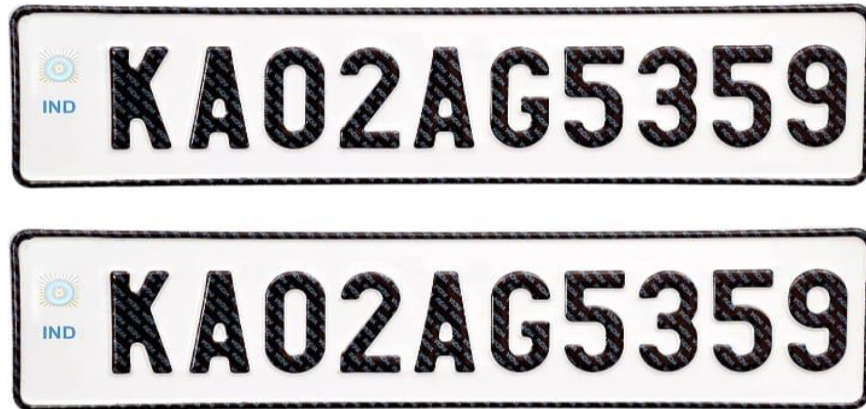


Fig. 1.7: Input data

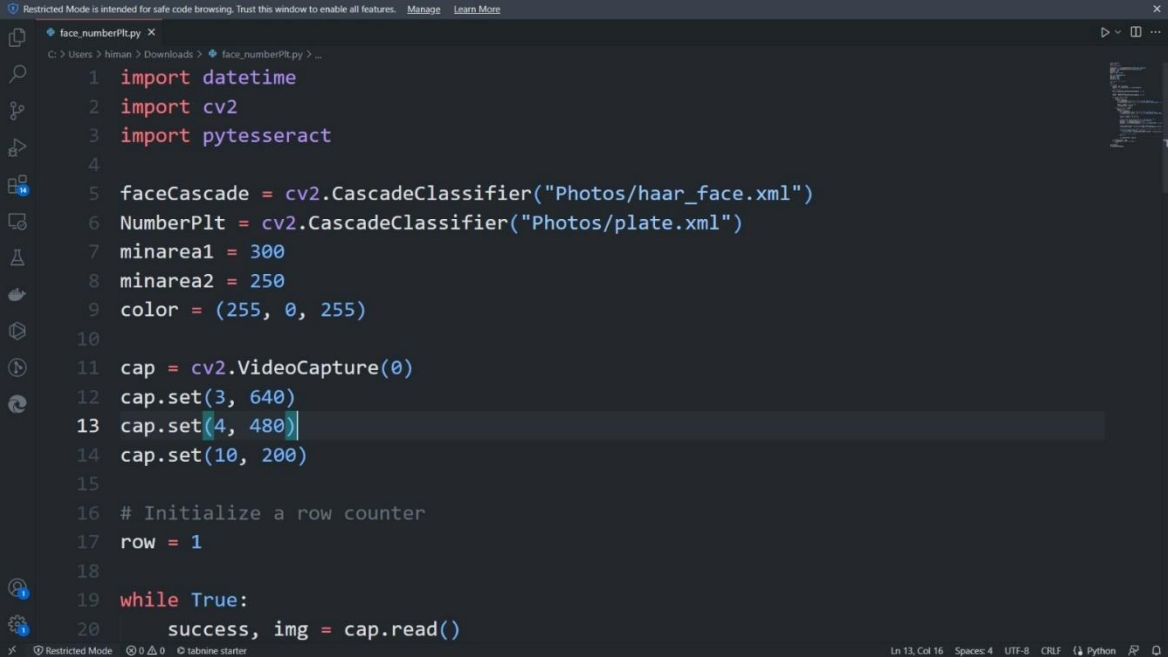
CHAPTER - 9

CONCLUSION

In this paper we have presented a real-time license plate detection and recognition system. The system operates on image frames acquired with standard video equipment without any additional sensor input. The high performance of the system allows for compensating the low image resolution by considering the classification results of subsequent frames. Due to the complete integration on an embedded device, the system operates autonomously, reporting only the final classification results to connected operators. Self-evidently, all advantages and capabilities of distributed systems apply here as well. We are currently working on an optimization to the detection algorithm implementation; our goal is to adapt the algorithm to better fit to the architectural characteristics of our platform. In the future we will also have a look at special region segmentation methods as the one proposed in [13] and examine their applicability on our hardware.

APPENDIX

SCREEN SHOTS



The screenshot shows a code editor window with a dark theme. The file name is 'face_numberPlt.py'. The code is a Python script for face detection and number recognition. It imports 'datetime', 'cv2', and 'pytesseract'. It initializes a face cascade classifier for 'Photos/haar_face.xml' and a number plate classifier for 'Photos/plate.xml'. It sets minimum area thresholds and a color. It then captures video from the default camera, sets resolution (640x480) and FPS (10), and enters a loop to read frames. The current line of code is 'success, img = cap.read()'. The status bar at the bottom indicates 'Ln 13, Col 16', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', and a search icon.

```
1 import datetime
2 import cv2
3 import pytesseract
4
5 faceCascade = cv2.CascadeClassifier("Photos/haar_face.xml")
6 NumberPlt = cv2.CascadeClassifier("Photos/plate.xml")
7 minarea1 = 300
8 minarea2 = 250
9 color = (255, 0, 255)
10
11 cap = cv2.VideoCapture(0)
12 cap.set(3, 640)
13 cap.set(4, 480)
14 cap.set(10, 200)
15
16 # Initialize a row counter
17 row = 1
18
19 while True:
20     success, img = cap.read()
```

```
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
face_numberPlt.py X
C:\Users\himan\Downloads> face_numberPlt.py > ...

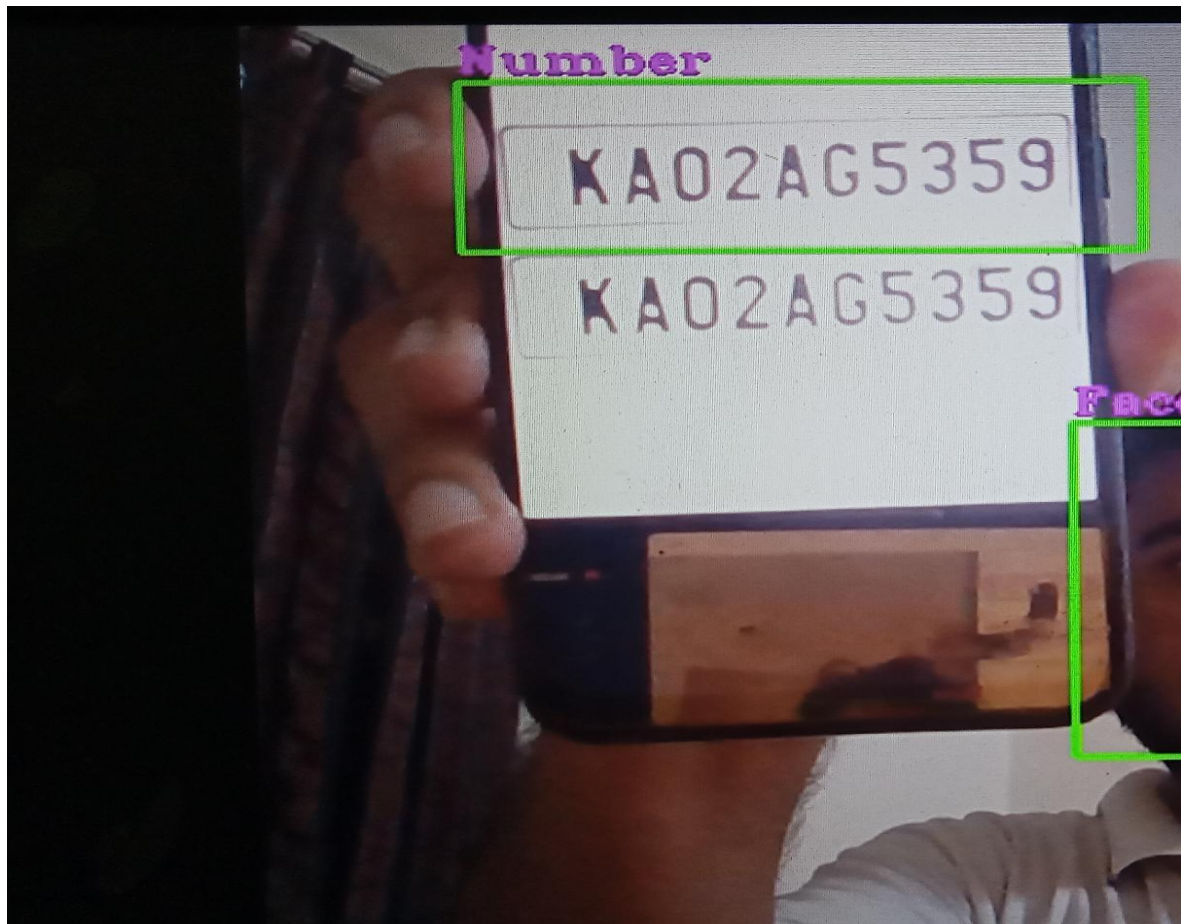
21 imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
22
23 # Detect faces
24 face = faceCascade.detectMultiScale(imgGray, 1.1, 5)
25
26 # Detect number plates
27 number = NumberPlt.detectMultiScale(imgGray, 1.1, 5)
28
29 for (x, y, w, h) in face:
30     area1 = w * h
31     if area1 > minarea1:
32         cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
33         cv2.putText(img, "Face", (x, y - 5), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1,
34
35         imgroi = img[y:y + h, x:x + w]
36         cv2.imshow("ROI", imgroi)
37
38     for (x, y, w, h) in number:
39         area2 = w * h
40         if area2 > minarea2:
```

```
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
face_numberPlt.py X
C:\Users\himan\Downloads> face_numberPlt.py > ...

41 cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
42 cv2.putText(img, "Number", (x, y - 5), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1,
43
44 # Crop the number plate image
45 img_roi = img[y:y + h, x:x + w]
46
47 # Apply image processing techniques to improve OCR accuracy
48 img_gray = cv2.cvtColor(img_roi, cv2.COLOR_BGR2GRAY)
49 img_gray = cv2.medianBlur(img_gray, 3)
50 img_gray = cv2.threshold(img_gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
51
52 # Use Tesseract to extract text from the number plate
53 license_plate_number = pytesseract.image_to_string(img_gray, config='--oem 4')
54
55 # Write the license plate number and the corresponding time to the file
56 with open('license_plates.txt', 'a') as f:
57     f.write(f"{row}. {license_plate_number.strip()} - {datetime.datetime.now()}\n")
58
59 # Increment the row counter
60 row += 1
```

```
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
face_numberPlt.py
C:\Users\himan\Downloads> face_numberPlt.py
50 img_gray = cv2.threshold(img_gray, 0, 255, cv2.THRESH_BINARY + cv2.T
51
52 # Use Tesseract to extract text from the number plate
53 license_plate_number = pytesseract.image_to_string(img_gray, config=
54
55 # Write the license plate number and the corresponding time to the t
56 with open('license_plates.txt', 'a') as f:
57     f.write(f"{row}. {license_plate_number.strip()} - {datetime.date
58
59 # Increment the row counter
60 row += 1
61
62 cv2.imshow("ROI2", img_roi)
63
64 cv2.imshow("Video", img)
65 if cv2.waitKey(1) & 0xFF == ord('q'):
66     break
67
68 cap.release()
69 cv2.destroyAllWindows()
```

```
2. KA02AG5359. - 2023-04-28 21:07:06
3. KAD2AS359 - 2023-04-28 21:07:07
4. KAOZAG5359 - 2023-04-28 21:07:08
5. KAOZAG5359 - 2023-04-28 21:07:09
6. KA02AG5359 - 2023-04-28 21:07:09
7. - 2023-04-28 21:07:10
8. | KAOZAG5359 - 2023-04-28 21:07:10
9. - 2023-04-28 21:07:10
10. | KAOZAG5359 - 2023-04-28 21:07:10
11. - 2023-04-28 21:07:11
12. KA02AG5359 - 2023-04-28 21:07:11
13. KA02AG5359 - 2023-04-28 21:07:12
14. KA02AG5359 - 2023-04-28 21:07:12
15. - 2023-04-28 21:08:51
```



CODE

```
import datetime
import cv2
import pytesseract

faceCascade = cv2.CascadeClassifier("Photos/haar_face.xml")
NumberPlt = cv2.CascadeClassifier("Photos/plate.xml")
minarea1 = 300
minarea2 = 250
color = (255, 0, 255)

cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
cap.set(10, 200)

# Initialize a row counter
row = 1

while True:
    success, img = cap.read()
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Detect faces
    face = faceCascade.detectMultiScale(imgGray, 1.1, 5)

    # Detect number plates
    number = NumberPlt.detectMultiScale(imgGray, 1.1, 5)
```

```

for (x, y, w, h) in face:
    area1 = w * h
    if area1 > minarea1:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(img, "Face", (x, y - 5),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, color, 2)

    imgroi = img[y:y + h, x:x + w]
    cv2.imshow("ROI", imgroi)

for (x, y, w, h) in number:
    area2 = w * h
    if area2 > minarea2:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(img, "Number", (x, y - 5),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, color, 2)

    # Crop the number plate image
    img_roi = img[y:y + h, x:x + w]

    # Apply image processing techniques to improve OCR
accuracy
    img_gray = cv2.cvtColor(img_roi, cv2.COLOR_BGR2GRAY)
    img_gray = cv2.medianBlur(img_gray, 3)
    img_gray = cv2.threshold(img_gray, 0, 255,
cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]

    # Use Tesseract to extract text from the number plate

```

```

        license_plate_number
pytesseract.image_to_string(img_gray, config='--psm 11')

    # Write the license plate number and the corresponding time
    to the text file
    with open('license_plates.txt', 'a') as f:
        f.write(f'{row}.        {license_plate_number.strip()}
{datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n')

    # Increment the row counter
    row += 1

    cv2.imshow("ROI2", img_roi)

cv2.imshow("Video", img)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()

```

REFERENCES

- <https://opencv.org/#>
- <https://pypi.org/project/Pillow/>
- <https://www.python.org/about/gettingstarted/>
- <https://numpy.org/learn>
- <https://opensource.google/tesseract>