

**UNREGISTERED VEHICLE RECOGNITION BASED ON NUMBER PLATE USING RASPBERRY PI**

**A PROJECT REPORT**

***Submitted by***

**JAYAVARSHINI V [REGISTER NO:211417104093] JOTHIKA B[REGISTER NO:211417104101]**

**KUSMITHAA S R[REGISTER NO:211417104127]**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.**

**ANNA UNIVERSITY: CHENNAI 600 025**

**APRIL 2021**

**BONAFIDE CERTIFICATE**

Certified that this project report **“UNREGISTERED VEHICLE RECOGNITION BASED ON NUMBER PLATE USING RASPBERRY PI”**is the bonafide work of **JAYAVARSHINI V (211417104093), JOTHIKA B(211417104101), KUSMITHAA SR(211417104127)**who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Dr.S.MURUGAVALLI,M.E.,Ph.D., M.SANGEETHA M.tech**

**HEAD OF THE DEPARTMENT SUPERVISOR**

**ASSOCIATE PROFESSOR**

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI, NAZARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on...........................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like express our heartfelt and sincere thanks to our Directors

**Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHIKUMAR, M.E., Ph.D.,** and

**Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide “M.SANGEETHA M.Tech ASSOCIATE PROFESSOR”** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

**JAYAVARSHINI V**

**JOTHIKA B**

**KUSMITHAA S R**

**ABSTRACT**

The automated object detection algorithm is really important component in the smart cities application to process the digitized images searching vehicle license plate. An USB camera is attached with raspberry pi as a processing unit to detect the number plates in traffic signals and image processing technology to search for a number plate in a given image frame. Extracted vehicle numbers are compared with existing database, if the number of vehicle is not in registered list it send the number and image of vehicle to the control station through mail.

The automated object detection algorithm is really important component in the smart cities application. In urban surveillance application the image sensor / camera plays an important role in digitizing the scene or environment. To process the digitized images searching for a particular object, smart vehicle license plate is a huge task as it will need a high CPU and memory power. To achieve this kind of functionality with distributing the processing is best way to solve. The Image processing technology to search for a number plate in a given image frame is an important task.

In this project we are using Raspberry pi as a processing unit to detect the number plates in traffic signals. USB camera is attached with raspberry pi, it detects number plates of every vehicle and extracts the numbers using OCR algorithm. Extracted vehicle numbers are compared with existing database, if the number of vehicle is not in registered list it send the number and image of vehicle to the control station through mail. Using this system, we can prevent illegal activities using unregistered or fake numbered vehicles. It is a low cost and efficient surveillance system when compared to present systems are used.

**LIST OF FIGURES**

**FIG. NO. DIAGRAM NAME PAGE NO.**

Fig.4.1 ER Diagram 20

Fig.4.2.1 Use Case diagram 21

Fig 4.2.2 Sequence diagram 23

Fig 4.2.3 Activity diagram 25

Fig 4.2.4 Collaboration diagram 27

Fig 5.1 System architecture 29

**LIST OF ABBREVIATIONS**

OCR -- Optical Character Resolution

LP -- License Plate

USB -- Universal Serial Bus

CNN -- Convolutional Neural Network

RNN -- Recurrent neural networks

CPU -- Central Processing Unit

UML -- Unified Modeling Language

ER -- Entity Relational

CCPD -- Certificate of Competency in Power Distribution

GPU -- Graphics Processing Unit

GCC -- GNU Compiler Collection

OS -- Operating System

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | iii |
|  | **LIST OF FIGURES** | viii |
|  | **LIST OF SYMBOLS, ABBREVIATIONS** | ix |
| **1.** | **INTRODUCTION** |  |
|  | 1.1 Overview |  |
|  | 1.2 Problem Definition |  |
| **2.** | **LITERATURE SURVEY** |  |
| **3.** | **SYSTEM ANALYSIS** |  |
|  | 2.1 Existing System |  |
|  | 2.2 Proposed system |  |
|  | 2.3 Requirement Analysis and Specification |  |
|  | 2.3.1 Input Requirements |  |
|  | 2.3.2 Output Requirements |  |
|  | 2.3.3 Functional Requirements |  |
|  | 2.4 Technology Stack |  |

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| **4.** | **SYSTEM DESIGN** |  |
|  | 4.1. ER diagram |  |
|  | 4.2 UML Diagrams |  |
| **5.** | **SYSTEM ARCHITECTURE** |  |
|  | 5.1 Architecture Overview |  |
| 5.2 Module Design Specification | | |
|  | | |
| **6.** | **SYSTEM IMPLEMENTATION** |  |
|  | 6.1 Client-side coding |  |
|  | 6.2 Server-side coding |  |
| **7.** | **SYSTEM TESTING** |  |
|  | 7.1 Unit Testing |  |
|  | 7.2 Integration Testing |  |
|  |  |  |
| **8.** | **CONCLUSION** |  |
|  | 8.1 Conclusion and Future Enhancements |  |
|  | **APPENDICES** |  |
|  | A.1 Sample Screens |  |

**REFERENCES**

**1. INTRODUCTION**

**1.1 Overview**

The automated object detection algorithm is really important component in the smart cities application. In urban surveillance application the image sensor / camera plays an important role in digitizing the scene or environment. To process the digitized images searching for a particular object, smart vehicle license plate is a huge task as it will need a high CPU and memory power. To achieve this kind of functionality with distributing the processing is best way to solve. The Image processing technology to search for a number plate in a given image frame is an important task.In this project we are using Raspberry pi as a processing unit to detect the number plates in traffic signals. USB camera is attached with raspberry pi, it detects number plates of every vehicle and extracts the numbers using OCR algorithm. Extracted vehicle numbers are compared with existing database, if the number of vehicle is not in registered list it send the number and image of vehicle to the control station through mail. Using this system, we can prevent illegal activities using unregistered or fake numbered vehicles. It is a low cost and efficient surveillance system when compared to present systems are used.

**1.2 Problem Definition**

In this paper, we tackle the problem of car license plate detection and recognition in natural scene images. We propose a unified deep neural network, which can localize license

plates and recognize the letters simultaneously in a single forward pass. The whole network can be trained end-to-end. In contrast to existing approaches which take license plate detection and recognition as two separate tasks and settle them step by step, our method jointly solves these two tasks by a single network. It not only avoids intermediate error accumulation but also accelerates the processing speed. For performance evaluation, four data sets including images captured from various scenes under different conditions are tested. Extensive experiments show the effectiveness and the efficiency of our proposed approach.

**2. LITERATURE SURVEY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SI.NO** | **YEAR** | **AUTHOR NAME** | **PROJECT TITLE** | **MERITS** | **DEMERITS** |
| 1 | 2015 | Baoguang Shi, Xiang Bai and Cong Yao | An End-to-End Trainable Neural Network for Image-based Sequence  Recognition and Its Application to Scene Text Recognition | The advantages of RNN is that it does not need the position of each element in a sequence object image in both training and testing.  A preprocessing step that converts an input object image into a sequence of image features, is usually essential. | The problem of scene text recognition, which is among the most important and challenging tasks in image-based sequence recognition |
| 2 | 2017 | Jiangmin Tian, Guoyou Wang | License plate detection in an open environment by density-based boundary clustering | The detection of boundary is simple and fast.  The more information about license plate is available | The main disadvantage is the variation of background, illumination, and view point, license plate detection in an open environment is challenging.    It is too time consuming for application with time requirement. |
| 3 | 2017 | Hui Li, Peng Wangy, and Chunhua Shen | Towards End-to-End Car License Plates Detection and Recognition with Deep Neural Networks | The advantage of end-to-end training for both detection and recognition in an unified network.  The sharing of convolutional features with both detection and recognition network, the model size decreases largely. | It does not explicitly handle the rotated plates.  Edge-based methods are fast in computation, but they cannot be applied to complex images  as they are too sensitive to unwanted edges. |
| 4 | 2016 | Chao Gou, KunfengWang, Yanjie Yao, and Zhengxi Li | Vehicle License Plate Recognition Based on Extremal Regions and Restricted Boltzmann Machines | The license plate by using a geometric template on connected target pixels with the same color.  These methods can detect license plate regions in relatively simple environments. | This may become invalid when there are regions in the image whose color information is similar to that of the license plate.  It can easily be affected by noises and are computationally complex when there are many edges in the image. |
| 5 | 2016 | Annie J. Zenath M.S, S. Shriram | Licence Plate Localization And Recognition Using Mser And Swt Algorithms | The neural network has advantage from existing correlation and statistics template techniques that allow being stable to noises and some position modifications of characters on license | If the training time is a critical factor then the network having single hidden layer.  The template matching has its drawbacks in some aspects comparing with neural networks. |
| 6 | 2016 | D. F. Llorca, C. Salinas, M. Jim´enez, I. Parra, A. G. Morcillo, R. Izquierdo, J. Lorenzo, M. A. Sotelo | Two-camera based accurate vehicle speed measurement using average speed at a fixed point | The system is run once the vehicle license plate is located by the first camera, and stopped once  it is not visible by the second one.The Highresolution and high-speed cameras with a narrow field . | The accurate speed detection of moving vehicles is a key issue to traffic law enforcement in most countries that may lead to an effective reduction in the number of road accidents and fatalities. It provides the minimum speed error independently of the speed of the vehicle. |
| 7 | 2016 | Jaderberg1 · Karen Simonyan1 · Andrea Vedaldi1 | Reading Text in theWild with Convolutional Neural Networks | An end-to-end system for text spotting—localizing and recognizing text in natural scene images—and text based image retrieval.This system is based on a region proposal mechanism for detection and deep convolutional neural networks for recognition. | This places text spotting as a separate, far more challenging problem than document OCR.This model was extended to CAPTCHA sequences up to 8 characters long where they demonstrated impressive performance using synthetic training data for a synthetic problem |
| 8 | 2017 | Zied Selmi, Mohamed Ben Halima | Deep Learning System for Automatic License Plate Detection and Recognition | The The detection and recognition of a vehicle License Plate (LP) is a key technique in most of the applications related to vehicle movement.It is a quite popular and active research topic in the field of image processing. | Some frameworks require complicated hardware to make good quality images or capture images from vehicles with very slow speed**.**Detection and recognition of LPs in different conditions and under several climatic variations remains always difficult to realize with good results |
| 9 | 2016 | Minghui Liao\_, Baoguang Shi\_, Xiang Baiy, Xinggang Wang, Wenyu | TextBoxes: A Fast Text Detector with a Single Deep Neural Network | TextBoxes outperforms competing methods in terms of text localization accuracy and is much faster, taking only 0.09s per image in a fast implementation.TextBoxes significantly outperforms state-of-the-art approaches on word spotting and end-to-end text recognition tasks | Owing to the inevitable challenges and complexities, traditional text detection methods tend to involve multiple processing steps.TextBoxes directly outputs the coordinates of word bounding boxes at multiple network layers by jointly predicting text presence and coordinate offsets to default boxes |
| 10 | 2018 | Zhenbo Xu | Towards End-to-End License Plate Detection and Recognition: A Large Dataset and Baseline | To our best knowledge, CCPD is the largest publicly available LP dataset to date with over 250k unique car images, and the only one provides vertices location annotations.  Through comparative experiments, we demonstrate our model outperforms current object detection and recognition approaches in both accuracy and speed. | An imperfect bounding box prediction given by detection methods might make a part of the LP missing, and thus results in the subsequent recognition failure.  Operations between different stages such as extracting and resizing the LP region for recognition are always accomplished by less efficient CPU, making LP recognition slower. |

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

Existing system classifies road signs for automated vehicle control. It uses cascade classifier to classify the road signs in roadside. It can label the name of road sign in live video. This system cannot give high accuracy in dynamic environments.For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation.

Baoguang Shi, Xiang Bai and Cong Yao-The problem of scene text recognition, which is among the most important and challenging tasks in image-based sequence recognition.

Jiangmin Tian, Guoyou Wang-The main disadvantage is the variation of background, illumination, and view point, license plate detection in an open environment is challenging.It is too time consuming for application with time requirement.

Hui Li, Peng Wangy, and Chunhua Shen-It does not explicitly handle the rotated plates.

Edge-based methods are fast in computation, but they cannot be applied to complex images

as they are too sensitive to unwanted edges.

Chao Gou, KunfengWang, Yanjie Yao, and Zhengxi Li-This may become invalid when there are regions in the image whose color information is similar to that of the license plate.It can easily be affected by noises and are computationally complex when there are many edges in the image

Annie J. Zenath M.S, S. Shriram- If the training time is a critical factor then the network having single hidden layer.The template matching has its drawbacks in some aspects comparing with neural networks.

D. F. Llorca, C. Salinas, M. Jim´enez, I. Parra, A. G. Morcillo, R. Izquierdo, J. Lorenzo, M. A. Sotelo-The accurate speed detection of moving vehicles is a key issue to traffic law enforcement in most countries that may lead to an effective reduction in the number of road accidents and fatalities. It provides the minimum speed error independently of the speed of the vehicle

Jaderberg1 · Karen Simonyan1 Andrea Vedaldi1-This places text spotting as a separate, far more challenging problem than document OCR.This model was extended to CAPTCHA sequences up to 8 characters long where they demonstrated impressive performance using synthetic training data for a synthetic problem.

-Zied Selmi, Mohamed Ben Halima-Some frameworks require complicated hardware to make good quality images or capture images from vehicles with very slow speed.Detection and recognition of LPs in different conditions and under several climatic variations remains always difficult to realize with good results

Minghui Liao\_, Baoguang Shi\_, Xiang Baiy, Xinggang Wang, Wenyu-

Owing to the inevitable challenges and complexities, traditional text detection methods tend to involve multiple processing steps.TextBoxes directly outputs the coordinates of word bounding boxes at multiple network layers by jointly predicting text presence and coordinate offsets to default boxes.

Zhenbo Xu-An imperfect bounding box prediction given by detection methods might make a part of the LP missing, and thus results in the subsequent recognition failure.

Operations between different stages such as extracting and resizing the LP region for recognition are always accomplished by less efficient CPU, making LP recognition slower.

**3.2 Proposed system**

Here we are proposing a unique combination of technologies which were available in the market. We are proposing to do the image processing in the local environment. The images from the camera were processed near the camera itself and the results were published to the central server for further processing. The scope of this project is to present a cost effective viable solutions, so we will be implementing the system and technologies needed to process the image locally and CNN techniques used in detecting the number plate region.

This system proposes a compact and portable model to recognize the number plate of the vehicle. It will extract the numbers from the detected number plates using OCR algorithm and compare the extracted numbers with existing database which contains all registered vehicle numbers. In this block diagram, Raspberry Pi is connected with camera. Camera captures the all frames continuously. Once the image is read in python the image is converted to grey scale image.

This means the pixel value will be stripped off the color values and converted to the grayscale. The grayscale image is then processed to find out possible number plate area using edge detection technique. After edge detection, all high intensity pixels were scanned from left to right in the X axis and top to bottom for Y axis. The concentration of white pixels will give a fair knowledge of where the more number of edges were available in x-axis. This will give the approximate starting location of x-axis. Concentration of white pixels in y-axis is also scanned to get the approximate location of the y-axis. Due to the variable text or images available in the vehicle, we may have more than one region of interest of the number plate. The region of interest for the number plate area was detected from the previous module. We will try to fit the number plate area and ratio of width and height. If the possible number plates region is identified, the image is cropped for the dimension of the detected number plate. The number plate image is then processed with simple optical character recognition software to convert to a text. The extracted text is compared with existing database content. If the number is not in the database it will send the image and number of vehicle to the control station.

**3.3 Requirement Analysis and Specification**

**3.3.1 Input Requirements**

AUTOMATIC car license plate detection and recognition plays an important role in intelligent transportation systems. It has a variety of potential applications ranging from security to traffic control, and attracts considerable research attentions during recent years.However, most of the existing algorithms only work well either under controlled conditions or with sophisticated image capture systems. It is still a challenging task to read license plates accurately in an uncontrolled environment. The difficulty lies in the highly complicatedbackgrounds, like thegeneral text in shop boards, windows, guardrail or bricks, and random photographing conditions, such as illumination, distortion, occlusion or blurring. Previous work on license plate detection and recognition usually considers plate detection and recognition as two separate tasks, and solves them respectively by different methods.A single unified deep neural network is proposed, which can detect license plates from an image and recognize the labels all at once. The whole framework involves no heuristic processes, such as the use of plate colors or character space, and avoids intermediate procedures like character grouping or separation. It can be trained end-to-end, with only the image, plate positions and labels needed for training. The resulting system achieves high accuracy on both plate detection and letter recognition.

**3.3.2 Output Requirements**

However, the tasks of plate detection and recognition are highly correlated. Accurate bounding boxes obtained via detection methods can improve the recognition accuracy, while the recognition result can be used to eliminate false positives vice versa. Thus in this paper, we propose a unified framework to jointly tackle these two tasks at the same level. A deep neural network is designed, which takes an image as input and outputs the locations of license plates as well as plate labels simultaneously, with both high efficiency and accuracy. The convolutional features are shared by both detection and recognition, which leads to fewer parameters compared to using separated models. Moreover, with the joint optimization of both detection and recognition losses, the extracted features would have richer information. Experiments show that both detection and recognition performance can be boosted via using the jointly trained model.

**3.3.3 Functional Requirements**

We prove that the low level features can be used for both detection and recognition. The whole network can be trained end to end, without using any heuristic rule. To our knowledge, this is the first work that integrates both license plate detection and recognition into a single end-to-end trainable network and solves them at the same time. By integrating plate recognition directly into the detection pipeline, instead of addressing them by separate models, the resulting system is more efficient. With our framework, we do not need to crop the detected license plates from the input image and then recognize them by a separate network. The whole framework takes about 0*.*31 second for an input image of 600×600 pixels on a Titan X GPU. It should be note that although a number of methods has been proposed for both text detection and recognition in natural scene.With this innovation, some pre-processing, like character detection or character grouping, are eliminated, and the intermediate errors can be avoided. The learned features can be more discriminative and lead to a better performance.

**3.4 Technology Stack**

**Hardware Requirements:**

Raspberry Pi 3

USB Camera

**Software Requirements:**

Language : Python

Compiler : GCC Complier.

OS :Linux

**Database**:

MySQL

**4. SYSTEM DESIGN**

**4.1. ER diagram**

USB CAMERA

EXTRACTS INFO

VEHICLE

Fig 4.1 ER Diagram

An **entity–relationship model** (or **ER model**) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between [entities](https://en.wiktionary.org/wiki/entity) (instances of those entity types).an ER model is commonly formed to represent things a business needs to remember in order to perform [business processes](https://en.wikipedia.org/wiki/Business_process). Consequently, the ER model becomes an abstract [data model](https://en.wikipedia.org/wiki/Data_modeling), that defines a data or information structure which can be implemented in a [database](https://en.wikipedia.org/wiki/Database), typically a [relational database](https://en.wikipedia.org/wiki/Relational_database). Entity–relationship modeling was developed for database and design by [Peter Chen](https://en.wikipedia.org/wiki/Peter_Chen) and published in a 1976 paper, with variants of the idea existing previously. Some ER models show super and subtype entities connected by generalization-specialization relationships, and an ER model can be used also in the specification of domain-specific [ontologies](https://en.wikipedia.org/wiki/Ontology_(computer_science)).From the above diagram we infer that the entity vehicle has attribute such as vehicle number and vehicle image.From the entity USB camera with its attributes resolution , size, brand,camera no. These two entities are related by extracts info.

|  |  |
| --- | --- |
|  |  |
|  |  |

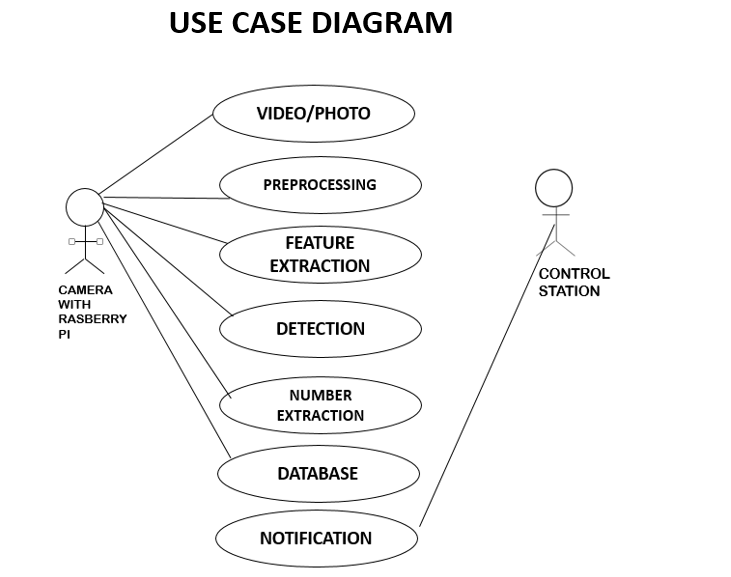
**4.2 UML Diagram**

Fig 4.2.1 Use Case Diagram

A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different [use cases](https://en.wikipedia.org/wiki/Use_case) in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.The purpose of use case diagram is to capture the dynamic aspect of a system. Additional diagrams and documentation can be used to provide a complete functional and technical view of the system.They provide the simplified and graphical representation of what the system must actually do.It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior. A use case **diagram** is usually simple. It does not show the detail of the use cases: It only summarizes some of the relationships between use cases, actors, and systems. 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 Statechart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.When the initial task is complete, use case diagrams are modelled to present the outside view.From the above diagram we infer that the user USB camera with raspberry pi performs capturing photo/video , preprocessing , feature extraction,number detection, number extraction and handles the database. Following the user control station receives notification through mail.

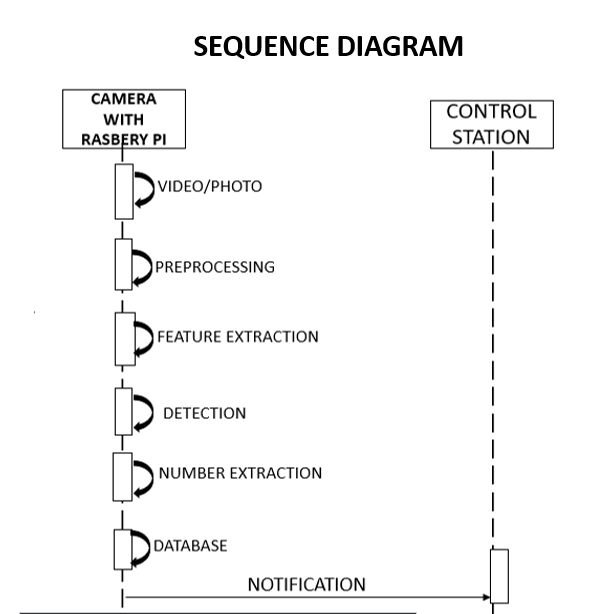


Fig 4.2.2 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the [Logical View](https://en.wikipedia.org/wiki/4%2B1_architectural_view_model) of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time.A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.From the above diagram we infer that the object USB camera with raspberry pi performs capturing photo/video , preprocessing , feature extraction,number detection, number extraction and handles the database. Following the object control station receives notification through mail.

**ACTIVITY DIAGRAM**

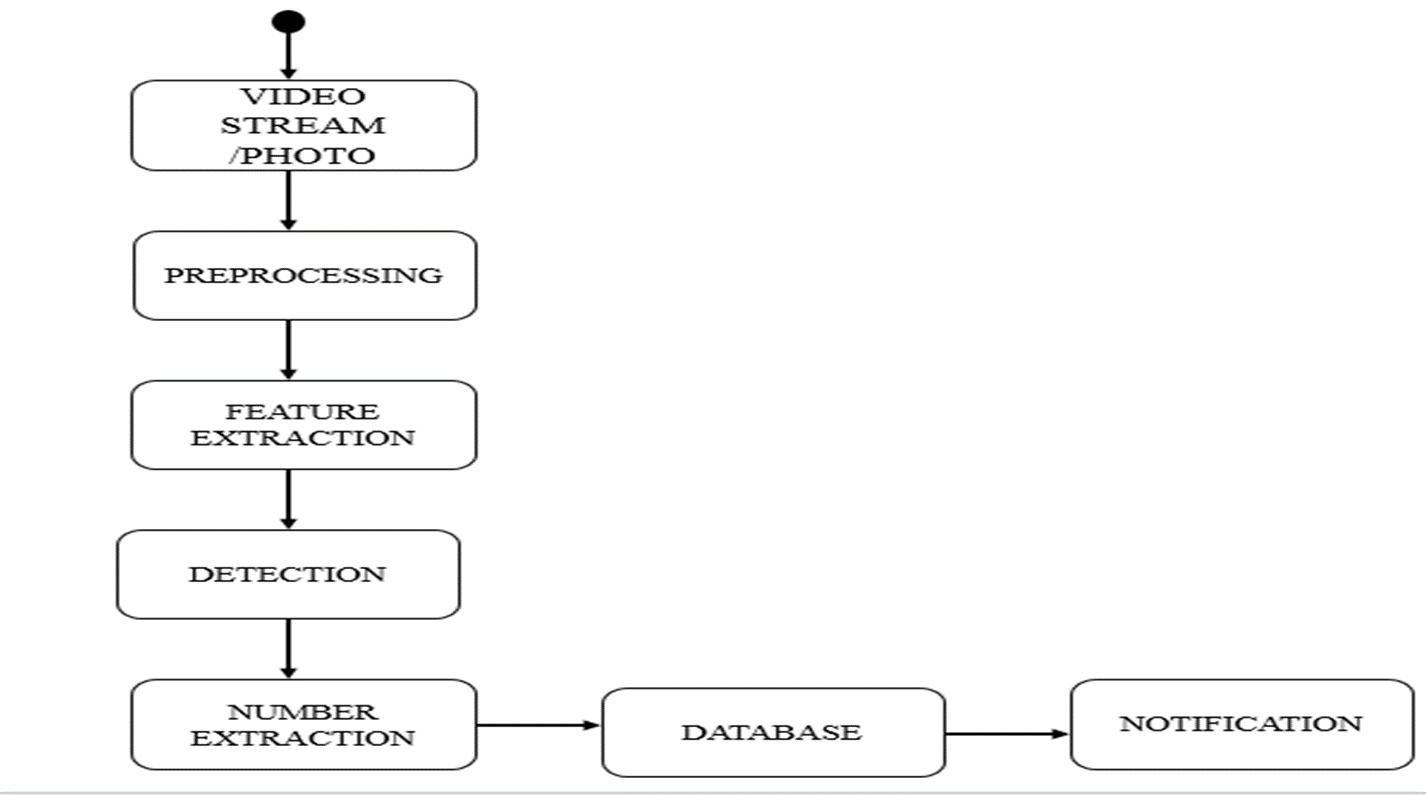


Fig 4.2.3 Activity Diagram

**Activity diagrams** are graphical representations of [workflows](https://en.wikipedia.org/wiki/Workflow) of stepwise activities and actions with support for choice, iteration and concurrency. In the [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language), activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities.Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.The basic purposes of activity diagrams is 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.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.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.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.From the above diagram we infer that the USB camera with raspberry pi performs captures photo/video , preprocessing , feature extraction,number detection, number extraction and handles the database. Following the control station receives notification through mail.

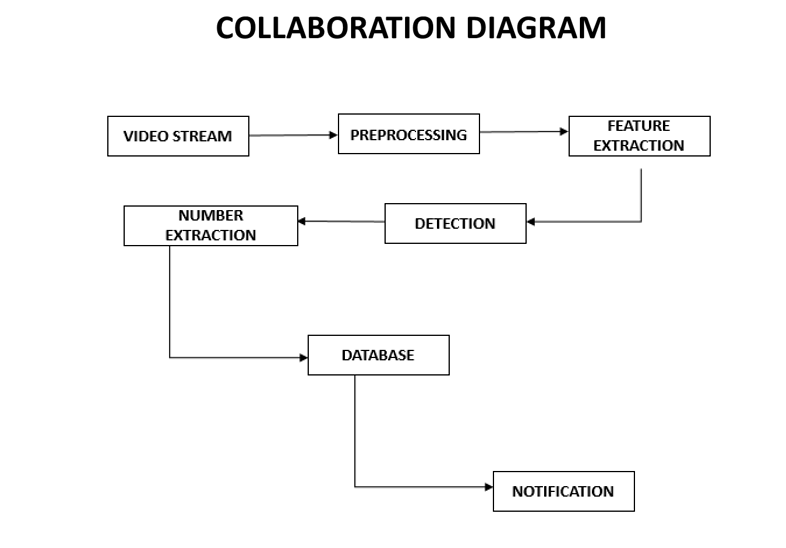


Fig.4.2.4 Collaboration Diagram

1The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.A **collaboration diagram**, also known as a communication **diagram**, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These **diagrams** can be used to portray the dynamic behavior of a particular use case and **define** the role of each object.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 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.Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.From the above diagram we infer that the USB camera with raspberry pi performs captures photo/video , preprocessing , feature extraction,number detection, number extraction and handles the database. Following the control station receives notification through mail

**5. SYSTEM ARCHITECTURE**

**5.1 Architecture Overview**

Diagram

Description automatically generated

**FIG 5.1.System architecture**

This system proposes a compact and portable model to recognize the number plate of the vehicle. It will extract the numbers from the detected number plates using OCR algorithm and compare the extracted numbers with existing database which contains all registered vehicle numbers.In this block diagram, Raspberry Pi is connected with camera. Camera captures the all frames continuously. Once the image is read in python the image is converted to grey scale image.This means the pixel value will be stripped off the color values and converted to the grayscale. The grayscale image is then processed to find out possible number plate area using edge detection technique.After edge detection, all high intensity pixels were scanned from left to right in the X axis and top to bottom for Y axis. The concentration of white pixels will give a fair knowledge of where the more number of edges were available in x-axis. This will give the approximate starting location of x-axis.Concentration of white pixels in y-axis is also scanned to get the approximate location of the y-axis. Due to the variable text or images available in the vehicle, we may have more than one region of interest of the number plate.The region of interest for the number plate area was detected from the previous module. We will try to fit the number plate area and ratio of width and height. If the possible number plates region is identified, the image is cropped for the dimension of the detected number plate. The number plate image is then processed with simple optical character recognition software to convert to a text. The extracted text is compared with existing database content. If the number is not in the database it will send the image and number of vehicle to the control station

**5.2 MODULE DESIGN SPECIFICATION**

* **Image Capturing**
* **Image Frame cleaning and preprocessing**
* **Detecting number plate using contours**
* **Transforming Number plate image to text**

**5.2.1 Image Capturing:**

In this module we create a basic infrastructure to establish the proposed feature of image handling in python. Prepare the python environment to access video input device, in this case a camera.

**5.2.2 Image Frame cleaning and preprocessing:**

Once the image is read in python we use different image processing technique on the image. The number plate detection technique is deployed. First the image is converted to grey scale image. This means the pixel value will be stripped off the color values and converted to the grayscale. The pixel will have a value of 1byte per pixel which means any value between 0-255 is used. The grayscale image is then processed to find out possible number plate area using edge detection technique.

**5.2.3 Detecting number plate using contours:**

After edge detection, all high intensity pixels were scanned from left to right in the X axis and top to bottom for Y axis. The concentration of white pixels will give a fair knowledge of where the more number of edges were available in x-axis. This will give the approximate starting location of x-axis. Concentration of white pixels in y-axis is also scanned to get the approximate location of the y-axis. Due to the variable text or images available in the vehicle, we may have more than one region of interest of the number plate.

**5.2.4 Transforming Number Plate Image to Text:**

The region of interest for the number plate area was detected from the previous module. We will try to fit the number plate area and ratio of width and height. If the possible number plates region is identified, the image is cropped for the dimension of the detected number plate. Then the CNN algorithm is performed. The number plate image is then processed with simple optical character recognition software to convert to a text.

**6. SYSTEM IMPLEMENTATION**

**6.1 Client-side coding**

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.base import MIMEBase

from email.mime.text import MIMEText

from email.utils import formatdate

from email import encoders

def sndMail(no):

toaddr = 'iotcloudsfire@gmail.com' # To id

me = 'iotcloudsfire@gmail.com' # your id

subject = "Unregistered CAR" # Subject

msg = MIMEMultipart()

msg['Subject'] = subject

msg['From'] = me

msg['To'] = toaddr

msg.preamble = "test "

msg.attach(MIMEText('Stolen CAR no: '+no+' Identified'))

part = MIMEBase('application', "octet-stream")

part.set\_payload(open("abc1.jpg", "rb").read())

encoders.encode\_base64(part)

part.add\_header('Content-Disposition', 'attachment; filename="saved\_img.jpg"') # File name and format name

msg.attach(part)

try:

s = smtplib.SMTP('smtp.gmail.com', 587) # Protocol

s.ehlo()

s.starttls()

s.ehlo()

s.login(user = 'iotcloudsfire@gmail.com', password = 'cloudiotcloud') # User id & password

#s.send\_message(msg)

s.sendmail(me, toaddr, msg.as\_string())

s.quit()

#except:

# print ("Error: unable to send email")

except SMTPException as error:

print ("Error") # Exception

**6.2 Server-side coding**

import re

import numpy as np

import cv2

from copy import deepcopy

from PIL import Image

import pytesseract as tess

import sys

import imutils

from imutils.video import VideoStream

from imutils.video import FPS

import time

from sen\_mail import sndMail

with open ("blacklist.csv") as f:

content=f.readlines()

content = [x.strip() for x in content]

print(content)

print(type(content))

img1=""

font = cv2.FONT\_HERSHEY\_SIMPLEX

bottomLeftCornerOfText = (10,50)

fontScale = 1

fontColor = (255,255,255)

redFontColor = (0,0,255)

lineType = 2

def preprocess(img):

imgBlurred = cv2.GaussianBlur(img, (5,5), 0)

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

sobelx = cv2.Sobel(gray,cv2.CV\_8U,1,0,ksize=3)

ret2,threshold\_img = cv2.threshold(sobelx,0,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

return threshold\_img

def cleanPlate(plate):

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

\_, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH\_BINARY)

contours,hierarchy = cv2.findContours(thresh.copy(),cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_NONE)

if contours:

areas = [cv2.contourArea(c) for c in contours]

max\_index = np.argmax(areas)

max\_cnt = contours[max\_index]

max\_cntArea = areas[max\_index]

x,y,w,h = cv2.boundingRect(max\_cnt)

img1=plate

if not ratioCheck(max\_cntArea,w,h):

return plate,None

cleaned\_final = thresh[y:y+h, x:x+w]

return img1,[x,y,w,h]

else:

return plate,None

def extract\_contours(threshold\_img):

element = cv2.getStructuringElement(shape=cv2.MORPH\_RECT, ksize=(17, 3))

morph\_img\_threshold = threshold\_img.copy()

cv2.morphologyEx(src=threshold\_img, op=cv2.MORPH\_CLOSE, kernel=element, dst=morph\_img\_threshold)

contours, hierarchy= cv2.findContours(morph\_img\_threshold,mode=cv2.RETR\_EXTERNAL,method=cv2.CHAIN\_APPROX\_SIMPLE)

return contours

def ratioCheck(area, width, height):

ratio = float(width) / float(height)

if ratio < 1:

ratio = 1 / ratio

aspect = 4.7272

min = 10\*aspect\*10

max = 125\*aspect\*125 # maximum area

rmin = 3

rmax = 6

if (area < min or area > max) or (ratio < rmin or ratio > rmax):

return False

return True

def isMaxWhite(plate):

avg = np.mean(plate)

if(avg>=95):

return True

else:

return False

def validateRotationAndRatio(rect):

(x, y), (width, height), rect\_angle = rect

if(width>height):

angle = -rect\_angle

else:

angle = 90 + rect\_angle

if angle>15:

return False

if height == 0 or width == 0:

return False

area = height\*width

if not ratioCheck(area,width,height):

return False

else:

return True

def cleanAndRead(img,contours):

global last\_no

for i,cnt in enumerate(contours):

min\_rect = cv2.minAreaRect(cnt)

if validateRotationAndRatio(min\_rect):

x,y,w,h = cv2.boundingRect(cnt)

plate\_img = img[y:y+h,x:x+w]

if(isMaxWhite(plate\_img)):

clean\_plate, rect = cleanPlate(plate\_img)

if rect:

x1,y1,w1,h1 = rect

x,y,w,h = x+x1,y+y1,w1,h1

plate\_im = Image.fromarray(clean\_plate)

plate\_img = img[y:y+h,x:x+w]

cv2.imwrite("abc.jpg",plate\_img)

climg = cv2.imread("abc.jpg")

clean\_plate = cv2.resize(climg, None, fx=3.5, fy=3.5, interpolation=cv2.INTER\_CUBIC)

config = ('--tessdata-dir "." -l eng --oem 1 --psm 3')

text = tess.image\_to\_string(Image.open('abc.jpg'),config=config)

regex = re.compile('[^a-zA-Z0-9]')

text = regex.sub('', text)

#print(text)

if(len(text)==10):

if(text not in content):

print(text)

print("Black")

cv2.imwrite("abc1.jpg",frame)

cv2.putText(img,text,(x,(y-10)),font,fontScale,redFontColor,lineType)

img = cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)

if(last\_no!=text):

last\_no=text

sndMail(text)

else:

cv2.putText(img,text,(x,(y-10)),font,fontScale,fontColor,lineType)

img = cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)

return img

if \_\_name\_\_ == '\_\_main\_\_':

vs = VideoStream(src=0).start()

time.sleep(2.0)

fps = FPS().start()

last\_no=""

while True:

frame = vs.read()

frame = imutils.resize(frame, width=500)

threshold\_img = preprocess(frame)

contours= extract\_contours(threshold\_img)

img = cleanAndRead(frame,contours)

cv2.imshow("Frame", img)

key = cv2.waitKey(1) & 0xFF

if key == ord("q"):

break

**7. SYSTEM TESTING**

**7.1 Unit Testing**

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing

**7.2 Integration Testing**

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

**TEST REPORT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TEST CASE NO. | TEST CASE DESCRIPTION | TEST DATA | EXPECTED RESULT | ACTUAL RESULT | PASS/FAIL |
| 1 | Extract the correct number from the image through USB camera and identify as registered vehicle. | A vehicle number plate shown in the USB camera. | The number plate should be recognized as KL 13 AH 1392  (registered vehicle) | Number plate recognized as KL 13 AH 1399  (unregistered vehicle) | Fail |
| 2 | Extract the correct number from the image through USB camera and identify as registered vehicle. | A vehicle number plate shown in the USB camera. | The number plate should be recognized as TN 57 AK 1234  (registered vehicle) | Number plate recognized as  TN 57 AK 1234  (registered vehicle) | Pass |
| 3 | Extract the correct number from the image through USB camera and identify as unregistered vehicle. | A vehicle number plate shown in the USB camera. | The number plate should be recognized as HR 26 DK 8337  (unregistered vehicle) | Number plate recognized as  HR 26 DK 8337  (unregistered vehicle) | Pass |
| 4 | Extract the correct number from the image through USB camera and identify as unregistered vehicle. | A vehicle number plate shown in the USB camera. | The number plate should be recognized as MH 12 DE 1433  (unregistered vehicle) | Number plate recognized as MH 12 0E 1433  (unregistered vehicle) | Fail |
| 5 | Extract the correct number from the image through USB camera and identify as unregistered vehicle. | A vehicle number plate shown in the USB camera. | The number plate should be recognized as CG 04 MF 2250  (unregistered vehicle) | Number plate recognized as  CG 04 MF 2250  (unregistered vehicle) | Pass |

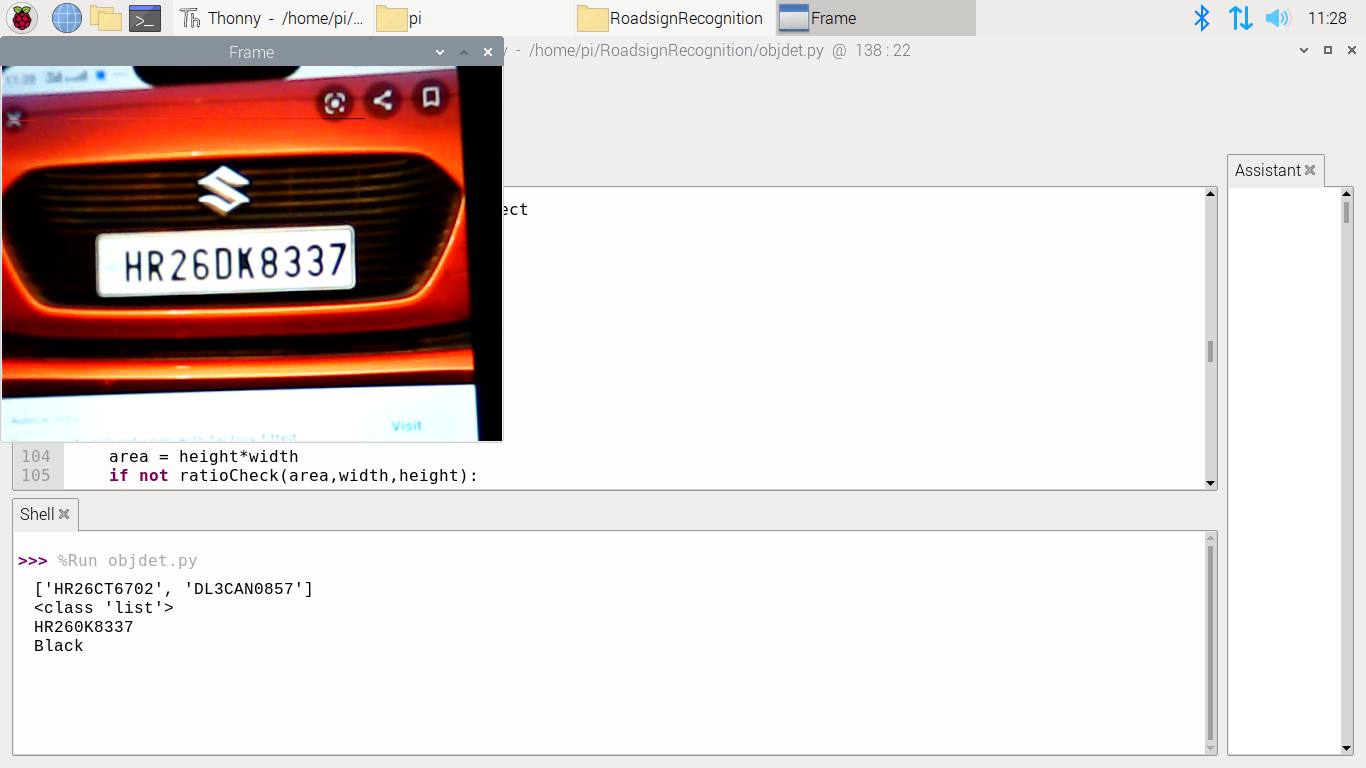
**8. CONCLUSION**

**8.1 Conclusion and Future Enhancements**

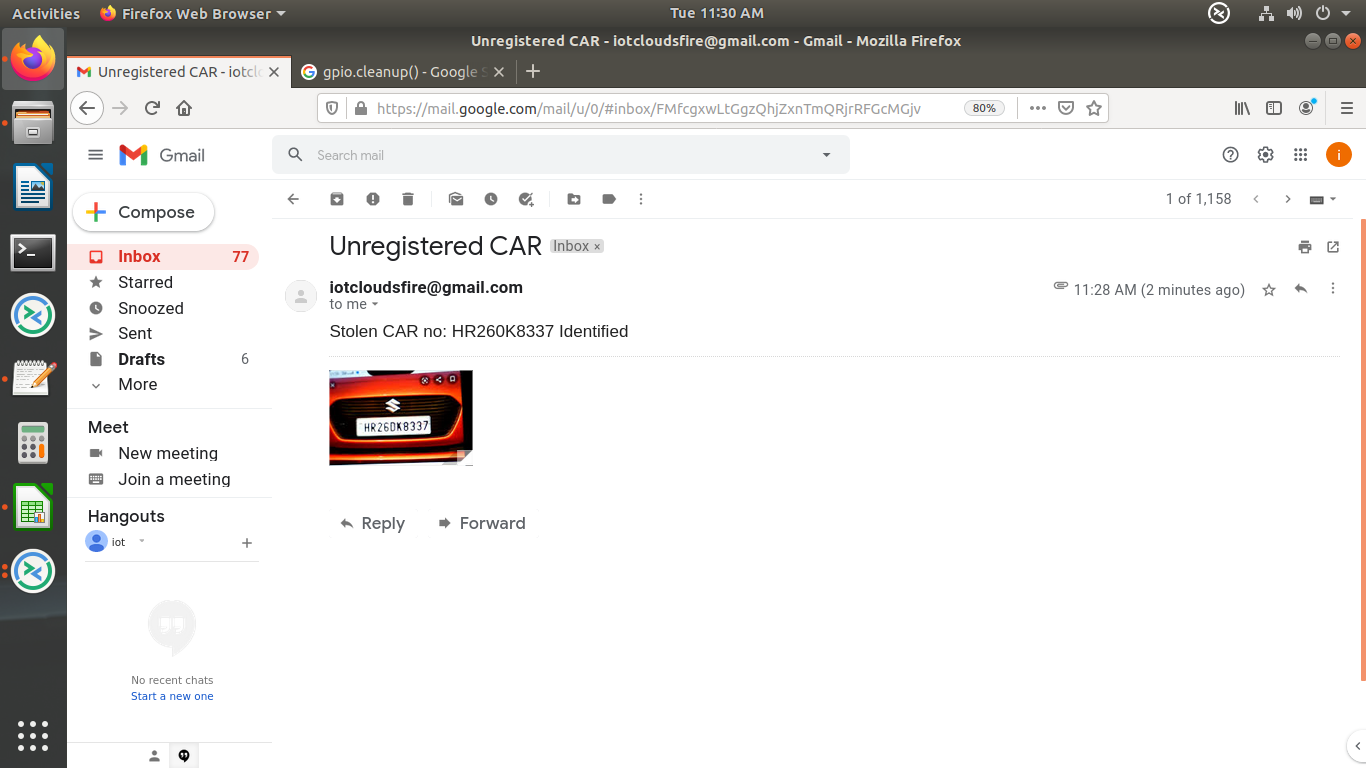
Thus in the unregistered vehicle recognition using raspberry pi we have overcome all the drawbacks till date. We have used the simplest algorithms and have reduced the time of the output. Our system also adapts to all the climatic conditions .Using this system, we can prevent illegal activities using unregistered or fake numbered vehicles .It is a low cost and efficient surveillance system when compared to present systems are used . This system is compact and portable model.

**APPENDICES**

**A.1 Sample Screens**

****

****

****

**REFERENCES**

[1] L. Neumann and J. Matas, “A method for text localization and recognition

in real-world images,” in *Proc. Asian Conf. Comput. Vis.*, 2011,

pp. 770–783.

[2] L. Neumann and J. Matas, “Real-time scene text localization and recognition,”

in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012,

pp. 3538–3545.

[3] K. Wang, B. Babenko, and S. Belongie, “End-to-end scene text recognition,”

in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 1457–1464.

[4] T. Wang, D. Wu, A. Coates, and A. Y. Ng, “End-to-end text recognition

with convolutional neural networks,” in *Proc. IEEE Int. Conf. Pattern*

*Recognit.*, Nov. 2012, pp. 3304–3308.

[5] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven, “PhotoOCR:

Reading text in uncontrolled conditions,” in *Proc. IEEE Int. Conf.*

*Comput. Vis.*, Dec. 2013, pp. 785–792.

[6] M. Jaderberg, A. Vedaldi, and A. Zisserman, “Deep features for text

spotting,” in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 512–528.

[7] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “Reading

text in the wild with convolutional neural networks,” *Int. J. Comput.*

*Vis.*, vol. 116, no. 1, pp. 1–20, 2016.

[8] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, “TextBoxes: A fast text

detector with a single deep neural network,” in *Proc. Nat. Conf. Artif.*

*Intell.*, 2017, pp. 4161–4167.

[9] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, “Automatic license

plate recognition (ALPR): A state-of-the-art review,” *IEEE Trans.*

*Circuits Syst. Video Technol.*, vol. 23, no. 2, pp. 311–325, Feb. 2013.

[10] W. Zhou, H. Li, Y. Lu, and Q. Tian, “Principal visual word discovery for

automatic license plate detection,” *IEEE Trans. Image Process.*, vol. 21,

no. 9, pp. 4269–4279, Sep. 2012.

[11] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and

E. Kayafas, “A license plate-recognition algorithm for intelligent transportation

system applications,” *IEEE Trans. Intell. Transp. Syst.*, vol. 7,

no. 3, pp. 377–392, Sep. 2006.

[12] G.-S. Hsu, J.-C. Chen, and Y.-Z. Chung, “Application-oriented license

plate recognition,” *IEEE Trans. Veh. Technol.*, vol. 62, no. 2,

pp. 552–561, Feb. 2013.

[13] Y. Yuan, W. Zou, Y. Zhao, X. Wang, X. Hu, and N. Komodakis,

“A robust and efficient approach to license plate detection,” *IEEE Trans.*

*Image Process.*, vol. 26, no. 3, pp. 1102–1114, Mar. 2017.

[14] A. H. Ashtari, M. J. Nordin, and M. Fathy, “An iranian license plate

recognition system based on color features,” *IEEE Trans. Intell. Transp.*

*Syst.*, vol. 15, no. 4, pp. 1690–1705, Aug. 2014.

[15] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, “Automatic

license plate recognition,” *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 1,

pp. 42–53, Mar. 2004.

[16] S. Yu, B. Li, Q. Zhang, C. Liu, and M.-Q. H. Meng, “A novel license

plate location method based on wavelet transform and EMD analysis,”

*Pattern Recognit.*, vol. 48, no. 1, pp. 114–125, 2015.

[17] I. Giannoukos, C.-N. Anagnostopoulos, V. Loumos, and E. Kayafas,

“Operator context scanning to support high segmentation rates for real

time license plate recognition,” *Pattern Recognit.*, vol. 43, no. 11,

pp. 3866–3878, 2010.

[18] B. Li, B. Tian, Y. Li, and D. Wen, “Component-based license plate

detection using conditional random field model,” *IEEE Trans. Intell.*

*Transp. Syst.*, vol. 14, no. 4, pp. 1690–1699, Dec. 2013.

[19] D. F. Llorca *et al.*, “Two-camera based accurate vehicle speed measurement

using average speed at a fixed point,” in *Proc. 19th Int. Conf.*

*Intell. Transp. Syst.*, Nov. 2016, pp. 2533–2538.

[20] C. Gou, K. Wang, Y. Yao, and Z. Li, “Vehicle license plate recognition

based on extremal regions and restricted boltzmann machines,” *IEEE*

*Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1096–1107, Apr. 2016.

[21] Y. Hou, X. Qin, X. Zhou, X. Zhou, and T. Zhang, “License plate

character segmentation based on stroke width transform,” in *Proc. 8th*

*Int. Congr. Image Signal Process.*, Oct. 2015, pp. 954–958.

[22] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, “Multidigit

number recognition from street view imagery using deep convolutional

neural networks,” in *Proc. Int. Conf. Learn. Represent.*, 2014,

pp. 1–13.

[23] O. Bulan, V. Kozitsky, P. Ramesh, and M. Shreve, “Segmentationand

annotation-free license plate recognition with deep localization and

failure identification,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9,

pp. 2351–2363, Sep. 2017.

[24] H. Li and C. Shen. (2016). “Reading car license plates using

deep convolutional neural networks and LSTMs.” [Online]. Available:

https://arxiv.org/abs/1601.05610

[25] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and

J. Schmidhuber, “A novel connectionist system for unconstrained handwriting

recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31,

no. 5, pp. 855–868, May 2009.

[26] L. Neumann and J. Matas, “On combining multiple segmentations

in scene text recognition,” in *Proc. 12th Int. Conf. Document Anal.*

*Recognit.*, Aug. 2013, pp. 523–527.

[27] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network

for image-based sequence recognition and its application to scene text

recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11,

pp. 2298–2304, Nov. 2017.

[28] M. Busta, L. Neumann, and J. Matas, “Deep TextSpotter: An end-to-end

trainable scene text localization and recognition framework,” in *Proc.*

*IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2223–2231.

[29] K. Simonyan and A. Zisserman, “Very deep convolutional networks for

large-scale image recognition,” *CoRR*, vol. abs/1409.1556, Sep. 2014.

[30] O. Russakovsky *et al.*, “ImageNet large scale visual recognition challenge,”

*Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[31] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,”

in *Proc. IEEE Conf. Comput. Vis. Pattern. Recognit.*, Jul. 2017,

pp. 6517–6525. [Online]. Available: https://arxiv.org/abs/1612.08242

[32] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards realtime

object detection with region proposal networks,” in *Proc. Adv.*

*Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[33] Z. Zhong, L. Jin, S. Zhang, and Z. Feng, “DeepText: A unified

framework for text proposal generation and text detection in natural

images,” *CoRR*, vol. abs/1605.07314, May 2016.

[34] R. Girshick, “Faster R-CNN,” in *Proc. IEEE Int. Conf. Comput. Vis.*,

Dec. 2015, pp. 1440–1448.

[35] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural*

*Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[36] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,”

*CoRR*, vol. abs/1412.6980, Dec. 2014.

[37] (2003). *Caltech Plate Dataset*. [Online]. Available: http://www.vision.

caltech.edu/html-files/archive.html

[38] L. Dlagnekov, “Video-based car surveillance: License plate, make, and

model reconition,” M.S. thesis, Dept. Comput. Sci., Univ. California,

San Diego, CA, USA, 2015.

[39] (2013). *Benchmarks*. [Online]. Available: https://github.com/openalpr/

benchmarks

[40] D. Karatzas *et al.*, “ICDAR 2015 robust reading competition,” in *Proc.*

*Int. Conf. Document Anal. Recognit.*, Aug. 2015, pp. 1156–1160.

[41] B. Shi, X. Wang, P. Lyv, C. Yao, and X. Bai, “Robust scene text

recognition with automatic rectification,” in *Proc. IEEE Conf. Comput.*

*Vis. Pattern Recognit.*, Jun. 2016, pp. 4168–4176.

[42] J. Tian, G. Wang, J. Liu, and Y. Xia, “License plate detection in an open

environment by density-based boundary clustering,” *J. Electron. Imag.*,

vol. 26, no. 3, pp. 033017-1–033017-11, 2017.

[43] S. Kim, H. Jeon, and H. Koo, “Deep-learning-based license plate detection

method using vehicle region extraction,” *Electron. Lett.*, vol. 53,

no. 15, pp. 1034–1036, 2017.