

FINDING ROAD ABNORMALITIES USING CNN

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Electrical and Electronics Engineering

by

MADHAVAN S (38140901)

SURYA B (38140054)



**DEPARTMENT OF ELECTRICAL AND ELECTRONICS
ENGINEERING**

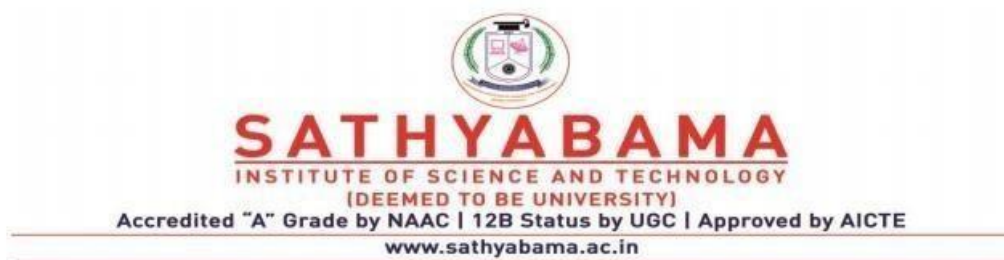
SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY 2022



DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **MADHAVAN S (38140901) SURYA B (38140054)** who carried out the project entitled "**FINDING ROAD ABNORMALITIES USING CNN**" under our supervision from December 2021 to May 2022

Internal Guide

Dr.S.Radhika

External guide

Dr. .V.SIVACHIDAMBARATHANATHAN M.E .,
Head of the Department

Submitted for Viva voce Examination held on_____

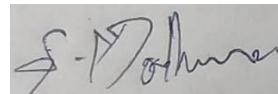
Internal Examiner

External Examiner

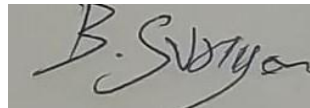
DECLARATION

We **MADHAVAN S** (38140901) and **SURYA B**(38140054) hereby declare that the Project Report entitled “**FINDING ROAD ABNORMALITIES USING CNN**” done by us under the guidance of **Dr.S.RADHIKA** at Sathyabama Institute of Science and Technology, **Chennai- 600119** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

1.



2.



DATE:

PLACE:

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

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

We convey our thanks to **Dr. N. M. NANDHITHA M.E., Ph.D., Dean, School of Electrical and Electronics Engineering** and **Dr.V.SIVACHIDAMBARATHANATHAN, Head of the Department, Dept. of Electrical and Electronics Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would also like to express our sincere and deep sense of gratitude to our internal guide **Dr.S.RADHIKA** for her valuable guidance and suggestions during the progressive reviews for our project.

ABSTRACT

Road potholes are common defects for aging civil infrastructures around the world. Furthermore, it is quite cost and labor intensive to repair potholes. The bad road conditions also lead to discomfort and even safety hazards to vehicle drivers or commuters, especially under high speed or low visibility conditions. Current researches are trying to use computer visions to detect the defects of the civil infrastructures, however the computer vision may fail in the night and need the expensive equipment. Therefore, this project designs and implements a portable Raspberry Pi system named Patrolman for more efficient and less-expensive road pothole inspection. The bad road conditions also lead to discomfort and even safety hazards to vehicle drivers or commuters, especially under high speed or low visibility conditions. Therefore, this project designs and implements a portable Raspberry Pi system named Patrolman for road pothole inspection.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	LIST OF TABLES	vi
	LIST OF FIGURES	viii
1.	INTRODUCTION	1
		2
	1.1 PURPOSE	
	1.2 PROJECT SCOPE	2
	1.3 EXISTING SYSTEM	4
	1.4 PROPOSED SYSTEM	4
	1.5 BENEFITS OF EXISTING SYSTEM	4
2.	LITERATURE SURVEY	6
3.	FUNDAMENTALS OF CNN	10
	3.1 INTRODUCTION	10
	3.2 FOUNDATION OF CNN	11
	3.3 CONCEPTS OF CNN	14

4.	DESIGN OF THE SYSTEM	15
	4.1 INTRODUCTION	15
	4.2 BLOCK DIAGRAM	15
	4.3 HARDWARE DESCRIPTION	16
	4.4 SOFTWARE DESCRIPTION	24
5.	RESULTS AND DISCUSSION	34
	5.1 Result	34
6	CONCLUSIONS AND FUTURE WORK	36
	6.1 CONCLUSION	36
	6.2 FUTURE WORK	36
	APPENDIX	37
	REFERENCES	44

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
4.1	BLOCK DIAGRAM	16
4.1	BLOCK DIAGRAM OF POWER SUPPLY	20
4.2	GENERAL BLOCK DIAGRAM OF IMAGE PREPROCESSING	23
4.3	BLOCK DIAGRAM	23
4.4	PUTTY LOGIN WINDOWS	25
4.5	PUTTY SUCCESSFUL WINDOWS	25
4.6	RASPBERRY PI ON LAPTOP DISPLAY	26
5.1	OUTPUT IMAGE	34
5.2	OUTPUT IMAGE	35

CHAPTER 1

INTRODUCTION

With the increase in number of vehicles on the roads and number of road networks, the issues related to road safety have become more pressing and complex. According to a World Health Organisations report, about 1.25 million people died in road accidents in 2015. Looking to data provided by Eurostat, only in the European Union, car accidents killed more than 25,000 people in 2016. Although Mohammed et al. did not find a direct impact of poor road condition on the number of accidents or accident related deaths, Teschke et al., Ahlin et al. showed that driving on roads with poor conditions negatively impacts the health of passengers. Concerning the identification of elements that characterise the road condition, besides potholes and other surface anomalies, speed bumps have also been identified as health hazards. In this context, providing road condition information to different stakeholders is an important task for driver safety, convenience and comfort. This may be achieved by constantly surveying roadsurface for anomalies and undertaking corrective measures repairing of roads or informing the stakeholders. Road pavement quality survey and anomalies detection is a useful service for various entities and different types of users. It is of interest to drivers of private as well as public transport vehicles. It is also important for public roads repair departments or other government authorities responsible for keeping the roads safe and in good condition.

The traditional methods for surveying the road condition are either by using specialised vehicles that have laser sensors or by surveying roads manually. However, these methods are very time Consuming and expensive, especially while maintaining a large network of roads in big cities. As an alternative, an emergent sensing paradigm, known as Collaborative Mobile Sensing or Mobile Crowdsourcing, is being discussed as a promising mechanism for collecting large-scale real-world data. Collaborative mobile sensing is a people-centred method of sensing, in which participants collect and contribute with sensing data, using mobile devices such as smartphones. The advantage of using smartphones is that it enables the participation of many contributors through crowdsourcing, thus enriching the system with a variety of data. Road anomaly detection or road condition survey systems

based on collaborative mobile sensing typically detect and automatically classify road anomalies using data-mining approaches on data collected by smartphones. This is a challenging task, especially when the data are collected in real-world deployments. The use of smartphones imposes several problems and limitations when compared with more expensive and specialized road-monitoring equipment. Specifically, smartphones use less expensive sensors that typically offer less precision when compared to specialized equipment. This is especially true for inertial and GPS sensors. This heterogeneity is not only related to smartphones and sensors. The automotive park suffers from the same problems, with many different car makers and models, and distinct suspension types and types. Furthermore, driving speed has a considerable impact on vehicle vibration, creating higher vibration amplitudes in accelerometer data. This work was carried out in the context of an industry–academia joint research and innovation project that aims at building road pavement quality services for drivers and government agencies. The objective was to evaluate a road anomalies detection system based on the data-mining approach described in a previous work, in a real world deployment, i.e., an environment in which data are collected during normal driving experience, which may originate some situations that were not perceived at the system design time. The data-mining approach in was based on the CRISP-DM framework that comprises the following phases: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation and Deployment. Smartphones were placed inside a car and a data collection application transmitted the data obtained from the smartphone sensors to a cloud-based platform, while the car was being driven. These “raw” data were then processed using various techniques to extract relevant features and machine-learning algorithms were applied for road anomalies recognition. Modelling and evaluation phases were based on a data set acquired. “Conditioned” setup, in which vehicles were moving on previously chosen road segments, with controlled speed. Our real-world deployment evaluation approach consisted of two phases. In the first phase, we analyzed the system’s performance in terms of the correct recognition of road anomalies in the real world deployment and compared the results with results obtained during the evaluation in the “conditioned” setup. We have found that there was a difference in the system’s performance, where the results obtained in the real world deployment were worse than the ones obtained in the previous evaluation phase. In the second phase, we aimed at identifying some issues that may have influenced the systems Performance. The

objective was threefold. First, in the context of the algorithm's training model, we intended to evaluate if training data sets that included sample generated from data collected through smartphones in the real world deployment would provide more robust models. Second, we analysed the complexity of the data as resulted from the data preparation phase of CRISP-DM framework. In particular, we executed some experiments to find out whether reducing or transforming data features would enhance the system performance. We based our features analysis on the Principal Component Analysis (PCA) technique. Third, considering the differences between the system's performance in each case, observed in the first phase, we provided an attribute-based analysis in the context of each anomaly type to better understand the characteristics of the acquired data and how those characteristics may influence in the distinction of different classes of anomalies.

The paper is structured as follows. A discussion on related work in the above domains is presented in Section 2. Section 3 briefly describes the approach for the road anomalies detection system,

1.1 PURPOSE

- Country's economy depend on well-maintained roads as they are major means of transportation. It becomes essential to identify pothole and humps in order to avoid accidents and damages to the vehicles that is caused because of distress to drivers and also to save fuel consumption.
- Controlling device used is Raspberry Pi. The system acquires the geographical position of potholes using Wi-Fi and transmits it to authorities to take corrective measures.

1.2 PROJECT SCOPE

- Design and Development of the Pothole and hump detection System.
- Help Road maintenance, aiding economic growth.
- Avoidance of accidents via real time alerts through webcam.
- To display the pothole location with attached webcam in the raspberry pi.

1.3 EXISTING SYSTEM

- The paper Uses a Kinect sensor, to obtain images from concrete and asphalt roads and exported to the MATLAB environment for processing. An extensive data processing algorithm analyses the image and the parameters like, pothole area, length, and width are estimated
- The vehicles and alert the vehicle user about the potholes via GSM SIM900A so the user can take precautionary mea
- Multiple digital image processing technologies, including: image preprocessing, binarization, thinning, three-dimensional reconstruction, error analysis and compensation are used in the series of image analysis and processing sure to avoid the accident.

1.4 PROPOSED SYSTEM

- Country's economy depend on well-maintained roads as they are major means of transportation. It becomes essential to identify pothole and humps in order to avoid accidents and damages to the vehicles that is caused because of distress to drivers and also to save fuel consumption.
- In this regard, this work presents a simple solution to detect potholes and humps and hence avoid accidents and help drivers. Potholes are detected using Image Processing Technique.
- Controlling device used is Raspberry Pi. The system acquires the geographical position of potholes using Wi-Fi and transmits it to authorities to take corrective measures.

1.5 BENEFIT OF PROPOSED SYSTEM

- This method will detect a problem before any problem accord and inform the driver and other passengers by raising an alert.
- In this OpenCV based machine learning techniques are used for automatic detection of potholes.
- Machine learning techniques takes less time to analyze the potholes.

1.6 PROBLEM STATEMENT

The goal of this proposal is to design and develop a prototype of an intelligent data gathering system that can be fitted in any transportation vehicle or car to detect the road or highway conditions.

- The prototype develops an information gathering system that informs the authorities about the road conditions.
- With this information the authorities could take measures to repair the roads which in turn help to avoid accidents caused due to nontechnical humps and hence lower the percentage of damaged roads and avoid road accidents.

CHAPTER 2

LITERATURE SURVEY

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it's worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully.

2.1 Road Anomaly Detection through Deep Learning Approaches

This paper addresses road anomaly detection by formulating it as a classification problem and applying deep learning approaches to solve it. Besides conventional road anomalies, additional ones are introduced from the perspective of a vehicle. In order to facilitate the learning process, the paper pays a close attention to pattern representation, and proposes three sets of numeric features for representing road conditions. Also, three deep learning approaches, i.e. Deep Feed forward Network (DFN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN), are considered to tackle the classification problem. The detectors, with respect to the three deep learning approaches, are trained and evaluated through data collected from a test vehicle driven on various road anomaly conditions. The comparison study on the detection performances is conducted by setting key hyper-parameters to certain sets of fixed values. Also, the comparison study on performances of each detector with respect to different pattern representations is conducted. The results have shown the effectiveness of the proposed approaches and the efficiency of the proposed feature representations in road anomaly detection.

2.2 Road Anomalies Detection System Evaluation

Anomalies on road pavement cause discomfort to drivers and passengers, and may cause mechanical failure or even accidents. Governments spend millions of Euros every year on road maintenance, often causing traffic jams and congestion on urban roads on a daily basis. This paper analyses the difference between the deployment of a road anomalies detection and identification system in a “conditioned”

and a real world setup, where the system performed worse compared to the “conditioned” setup. It also presents a system performance analysis based on the analysis of the training data sets; on the analysis of the attributes complexity, through the application of PCA techniques; and on the analysis of the attributes in the context of each anomaly type, using acceleration standard deviation attributes to observe how different anomalies classes are distributed in the Cartesian coordinates system. Overall, in this paper, we describe the main insights on road anomalies detection challenges to support the design and deployment of a new iteration of our system towards the deployment of a road anomaly detection.

Anomalies on road pavement cause discomfort to drivers and passengers, and may cause mechanical failure or even accidents. Governments spend millions of Euros every year in road maintenance, often causing traffic jams and congestion on urban roads on a daily basis.

In this paper, we describe a road anomalies detection system based on collaborative mobile sensing that detects and automatically classifies road anomalies using data-mining approaches on data collected by smartphone sensors, more specifically accelerometer data. We also present a service-oriented architecture that supports data acquisition, data processing, data storage, a road anomalies service and the development of end-user applications in the context of road condition domains. We deployed the road anomalies detection system in a real-world environment, in which driving activity was performed without any constraints of speed or road design. This was achieved using an end-user application for validating anomalies information, building a new ground-truth to evaluate the model’s performance. In total, 382 events were validated.

Score, precision and accuracy results in the real world deployment are significantly lower than score, precision and accuracy results in the “conditioned” setup. This represents a decrease in the system’s performance in the real world, particularly in recognising road anomalies and in distinguishing road anomalies classes. In a subsequent analysis of the training data sets, when including the data acquired in a real-world setup in the training phase, the overall performance

(Score measure) of the system did not improve. Nevertheless, we can observe an improvement in system’s precision. This may be because real-world data sets may include bigger diversity of data for the same anomaly class, which may reflect real conditions. This result points out that work on characterization and standardization of

the data acquisition process for real anomaly detection is needed. Similarly, our attempt to reduce the complexity of the data set attributes through PCA techniques has not shown any improvement of the final results.

When focusing on the aspect of distinguishing types (classes) of anomalies, we produced a visual analysis of the interactions between acceleration standard deviation X and acceleration standard deviation Y for the instances of the chosen anomalies classes. In this analyses, the aforementioned attributes' effectiveness in detecting the anomaly (and "no anomaly") cases and some similarity between the aforementioned attributes' values for "long" and "short bumps" classes were observed.

This may indicate that the classification problem should be divided into two main steps: first detecting the existence of an anomaly and then identifying its class. Furthermore, current results may show that long and short bumps classes should be merged into a unique class.

Taking all these factors into consideration, the deployment of our system helped us in gaining more insights about road anomalies detection challenges. This work is the first iteration of an on-going development towards the deployment a road anomaly detection service to provide information about roads condition to drivers and government entities. Despite the differences in system's performance between "conditioned" and real-world setups, we have obtained some insights that should support the design and deployment of a new iteration of our system

.

2.3 Road Anomalies Detection Using Basic Morphological Algorithms

In this paper some approaches for pothole detection of roads, using morphological algorithms, are recalled and tested. For road anomalies detection, one of the key elements is the pavement pothole information. Any algorithm for pothole detection has certain advantages and limitations as well, due to the real world environment, which is highly unstructured and dynamic. For road segmentation, the road anomalies. There is a large interest in the automatic road detection in the last years. The costs of the road maintenance have risen considerably. The damaged pavement is increasing in some areas due to the climate change – heavy rains and snows, inefficient methods to drain the water for the roads, and of the increasing traffic as well.

One of the key elements for road anomalies detection is based on the pavement pothole information. A pothole is a type of failure in the pavement surface, caused by factors like erosion, weather, traffic, etc.

When such anomalies are accumulated in the transportation system, they may create some major problems as: accidents, traffic coagulations, inefficient engine functioning, damage to the vehicles that use those roads with potholes, etc. These problems have a large impact on the economy and on the day to day life of people. If the roads are verified timely, potholes can be localized and filled up. In this way the traffic system can be improved.

The detection of potholes on roads by the aid of intelligent systems is a well-studied problem [1]. The identification, and hence the avoidance of potholes, may reduce the fuel consumption, wear-tear, and maintenance cost of vehicles.

By avoiding potholes, road safety increases and, indirectly the total travel time can be decreased, in some situations. Some solutions for the above mention problem can be: establishing pothole hotlines, holding contests to report particularly bad potholes. And asking people to contribute with potholes' images.

CHAPTER 3

Fundamental Concepts of Convolutional Neural Network

Convolutional neural network (or CNN) is a special type of multilayer neural network or deep learning architecture inspired by the visual system of living beings. The CNN is very much suitable for different fields of computer vision and natural language processing. The main focus of this chapter is an elaborate discussion of all the basic components of CNN. It also gives a general view of foundation of CNN, recent advancements of CNN and some major application areas.

3.1 Introduction

Among different deep learning architecture, a special type of multilayer neural network for spatial data is Convolutional Neural Network (or CNN or ConvNet.). The architecture of CNN is inspired by the visual perception of living beings. Though it is become popular after the record breaking performance of AlexNet[20] in 2012 but it is actually initiated in 1980. After 2012, the CNN got the pace to take over different fields of computer vision, natural language processing and many more. The foundation of convolutional neural network started from the discovery of Hubel and Wiesel in 1959 [16]. According to them, cells of animal visual cortex recognize light in the small receptive field. In 1980, inspired by this work, Kunihiko Fukusima proposed neocognitron.. This network is considered as the first theoretical model for CNN. In 1990, LeCun et al. developed the modern framework of CNN called LeNet-5[21] to recognize handwritten digits. Training by backpropagation[35] algorithm helped LeNet-5 in recognizing visual patterns from raw images directly without using any separate feature engineering. But in those days despite of several merits, the performance of CNN in complex problems was lacked by the limited training data, lack of innovation in algorithm and insufficient computing power. Recently we have large labeled datasets, innovative algorithms and powerful GPU machines. In 2012, with these upgradation, a large deep CNN, called AlexNet, designed by Krizhevsky et al[20] showed excellent performance on the ILSVRC[36]. The success of AlexNet paved the way to invent different CNN models[41] as well as to apply those models in different fields of computer vision and natural language processing.

A traditional convolutional neural network is made up of single or multiple blocks of convolution and pooling layers, followed by one or multiple fully connected (FC) layers

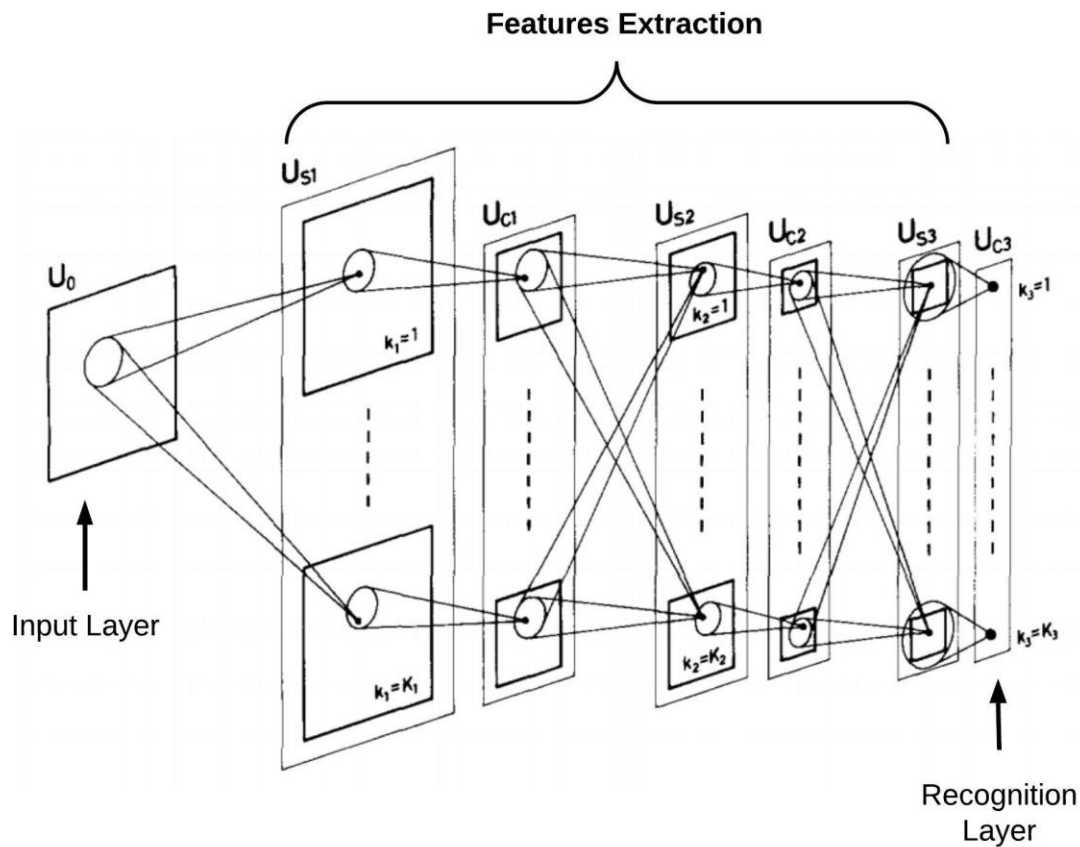
and an output layer. The convolutional layer is the core building block of a CNN. This layer aims to learn feature representations of the input. The convolutional layer is composed of several learnable convolution kernels or filters which are used to compute different feature maps. Each unit of feature map is connected to a receptive field in the previous layer. The new feature map is produced by convolving the input with the kernels and applying elementwise non-linear activation function on the convolved result. The parameter sharing property of convolutional layer reduces the model complexity. Pooling or sub-sampling layer takes a small region of the convolutional output as input and downsamples it to produce a single output. There are different sub-sampling techniques as example max pooling, min pooling, average pooling, etc. Pooling reduces the number of parameters to be computed as well as it makes the network translation

invariant. Last part of CNN is basically made up of one or more FC layers typically found in feedforward neural network. The FC layer takes input from the pooling or convolutional layer and generates output of CNN. In case of image classification, a CNN can be viewed as a combination of two parts: feature extraction part and classification part. Both convolution and pooling layers perform feature extraction. As an example of dog's image, different convolution layers from lower level to higher level detect various features such as two eyes, long ears, four legs, etc for further recognition. On top of this features, the FC layers are added as classifier, and a probability is assigned for the input image being a dog. Beside the layer design, the improvement of CNN depends on several different aspects such as activation function, normalization method, loss function, regularization, optimization and processing speed, etc. After the success of AlexNet, CNN got huge popularity in three major fields namely image classification[41], object detection[42] and segmentation[45], and many advance models of CNN has been proposed in those areas in successive years. Major application areas that apply CNN to achieve state-of-the-art performance includes image classification, object tracking, object detection, segmentation, human pose estimation, text detection, visual saliency detection, action recognition, scene labelling, visual question answering, speech and natural language processing, etc. Though current CNN models work very well for various applications, it is yet to know why and how it works essentially. So, more efforts on investigating the fundamental principles of CNNs are required.

3.2 Foundation of Convolutional Neural Network

In 1959, two neurophysiologists David Hubel and Torsten Wiesel experimented and later published their paper, entitled "single neurons in cat's striate cortex" described that the neurons inside the brain of a cat are organized in layered form.

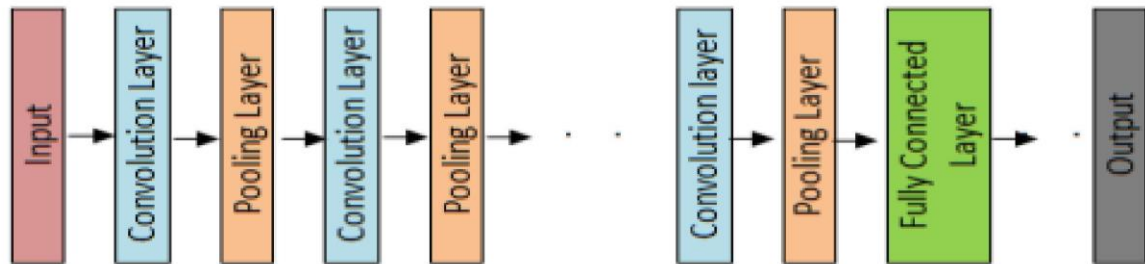
These layers learn how to recognize visual patterns by first extracting the local features and then combining the extracted features for higher level representation. Later on, this concept is essentially become one of the core principle of Deep Learning. Inspired by the work of Hubel and Wiesel, in 1980, Kunihiko Fukushima proposed Neocognitron, which is a self-organizing Neural Network, containing multiple layers, capable of recognizing visual patterns hierarchically through learning and this architecture became the theoretical model of CNN as in the . A major improvement over the architecture Neocognitron was done by LeCun et. In 1989 by developing a modern framework of CNN, called LeNet-5, which successfully recognized the MNIST handwritten digits dataset. LeNet-5 was trained using error back-propagation algorithm and it can be recognize visual patterns directly from raw input images, without using any separated feature engineering mechanism. After discovering LeNet-5, because of several limitation like lack of large training data, lack of innovation in algorithm and inadequate computing power, CNN did not performs well in various complex problems. But nowadays, in the era of Big Data we have large labeled datasets, more innovative algorithms and especially powerfull GPU machines. With these type of up-gradation, in 2012, Krizhevsky et al. designed AlexNet, which achieved a fantastic accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The victory of AlexNet paved the way to invent several CNN models as well as to apply those models in different eld of computer vision and natural language processing.



3.3 Concepts of Convolutional Neural Network

Convolutional Neural Network (CNN), also called ConvNet, is a type of Artificial Neural Network (ANN), which has deep feed-forward architecture and has amazing generalizing ability as compared to other networks with FC layers, it can learn highly abstracted features of objects especially spatial data and can identify them more efficiently.

A deep CNN model consists of a set of processing layers that can learn various features of input data (e.g. image) with multiple level of abstraction. The initiatory layers learn and extract the high level features (with lower abstraction), and the deeper layers learn and extract the low level features (with higher abstraction). The basic conceptual model of CNN was shown in _ different types of layers described in subsequent sections.



Conceptual model of CNN

- One of the main reason for considering CNN in such case is the weight sharing feature of
- CNN, that reduce the number of trainable parameters in the network, which helped the model to avoid overwriting and as well as to improved generalization.
- In CNN, the classification layer and the feature extraction layers learn together, that makes the output of the model more organized and makes the output more dependent to the extracted features.
- The implementation of a large network is more dicult by using other types of neural networks rather than using Convolutional Neural Networks.

CHAPTER 4

DESIGN OF THE SYSTEM

4.1 INTRODUCTION

We have described a prototype system to read printed text on hand-held objects for assisting blind persons. In order to solve the common aiming problem for blind users, we have proposed a motion-based method to detect the object of interest, while the blind user simply shakes the object for a couple of seconds. The automatic ROI detection and text localization algorithms were independently evaluated as unit tests to ensure effectiveness and robustness of the whole system. We subsequently evaluated this prototype system of assistive text reading using images of hand-held objects captured by ten blind users in person. Two calibrations were applied to prepare for the system test. First, we instructed blind users to place hand-held object within the camera view. Since it is difficult for blind users to aim their held objects, we employed a camera with a reasonably wide angle.

In future systems, we will add finger point detection and tracking to adaptively instruct blind users to aim the object. Second, in an applicable blind-assistive system, a text localization algorithm might prefer higher recall by sacrificing some precision

When our application starts running it first check all the devices and resources which it needs are available or not. After that it checks the connection with the devices and gives control to the user. The GUI for the user has the following options. An optional label is used for displaying the image taken from the camera. A status box is for representing the detected data from the image. The capture button is to detect the data from the image. The detect button is to detect the human from the video streaming in front of the camera. The audio jack port is the output port here. The Raspberry board comes with integrated peripherals like USB, ADC and Serial etc. On this board we are installing Linux operating system with necessary drivers for all peripheral devices.

Hardware

- Raspberry Pi

- Keyboard Mouse
- Monitor
- Speaker

Software

- Raspian OS
- Python
- Open CV

4.2 BLOCK DIAGRAM:

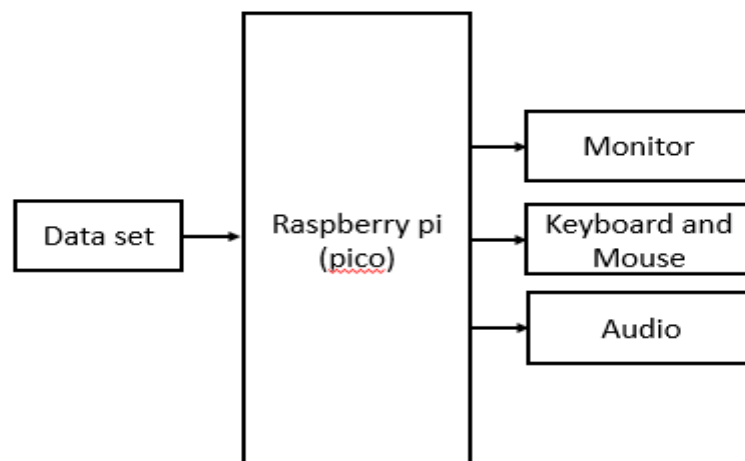


FIG 4.1 BLOCK DIAGRAM

4.3 HARDWARE DESCRIPTION

RASPBERRY PI

Overview

Raspberry Pi seems to be new in the world and many people really don't know what the Raspberry Pi is. Raspberry Pi can be defined as a low cost, credit-card

sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you would expect a desktop computer to do, from browsing the internet and playing high definition video, to making spreadsheets, word-processing, and playing games. It is great bonding with Arduino and can do a lot with Arduino.

Exploring the Raspberry Pi Board

There are two models of Raspberry Pi, model A and model B. These two are bit similar with few advance features on model B compared to model A. Model B has 512 MB RAM, two USB port where as Model A has 256 MB RAM and just a USB port. Besides, Model B has Ethernet port while Model A does not.

Figure 5-1: Raspberry Pi Model B(Sanders, 2013)

SD Card Slot

Raspberry Pi doesn't have the real hard drive as in laptop and computer, SD card is taken as solid state drive (SSD) which is used to install operating system and all others software and store everything. This card is needed to insert into the slot for using the Raspberry Pi.

SD card may be 2GB, 4GB or 16GB.

Micro USB Power

The power port is a 5V micro-USB input and supply should be exactly 5v as it doesn't have onboard power regulator. So, power supply shouldn't exceed than 5V.

HDMI Out

This output port is used to connect the Raspberry Pi with a monitor via HDMI (High Definition Multimedia Interface). Hence, any screen or TV can be connected to it which consists of HDMI port.

Ethernet and USB port

Both the Ethernet port and USB port on Model B are supplied via the onboard LAN9512 chip. It is a high-speed USB 2.0 hub with a 10/100 Ethernet controller (Donat, 2014). USB ports are used to connect the inputs (keyboard, mouse). Almost everything that can connect to computer via USB also can connect with Raspberry Pi.

RCA Video Out and Audio Out

Audio and RCA video jacks are present on the board for audio and video out. The Raspberry Pi does support sound over its HDMI output, but there is a standard 3.5-mm audio jack to plug in headphones but USB mikes may work or not.

For video, the RCA jack sends video to any connected RCA video device.

GPIO Headers(Pins)

GPIO pins stands for general purpose of input output pins. These pins are used to connect any number of physical extensions with the Raspberry Pi. Raspberry Pi has pre-installed libraries that allow us to access the pins using programming languages like C, C++ or python.

Chips (Broadcom)

The most important component in a Raspberry Pi is chip that is Broadcom which is placed at the middle of the board. The chip consists of ARM11 processor running at 700 MHz and a Videocore4 GPU and can be over clocked to at least 800 MHz without a problem.

Raspberry Pi 2 Model B

Recently, Raspberry Pi 2 Model B has been launched recently which Broadcom BCM2836 ARM Cortex-A7 Quad Core Processor has powered Single Board Computer running at 900MHz, 1GB RAM and 4 Quad USB ports. It is the advanced version of Model B and is 6 times faster than Model B Raspberry Pi. In addition, it has combined 4-pole jack for connecting your stereo audio out and composite video out and advanced power management. Figure 5-2 shows the top view of the board with labels of some important components (raspberrypi.org, 2015a).

Hardware Required for Raspberry Pi

Raspberry Pi can't start alone, it needs many others peripherals (hardware). There is brief description of the hardware requirements in the following section (Bates, 2014).

Power Supply

As mentioned already in above theory portion, Raspberry Pi needs 5V power supply. If supply exceeds 5V then it can't guaranteed to work properly. And the power supply also need to supply at least 500 milliamps (mA), and preferably more like 1 amp (A). If the supply is 500 mA or less, it is likely to have the mal-function of keyboard and mouse. It is not good idea to power the Raspberry Pi from USB port of computer and hub as they mostly provide current less than required. Hence, the Raspberry Pi requires a Micro-USB connection which is capable of supplying at least 700 mA (or 0.7 A) at 5V.

Storage

A separate hardware is required for the storage purpose in Raspberry Pi. For this, SD card is used, mostly 4 GB and 8 GB if needed. The operating system and all files are stored in the card. We can buy blank SD card and install operating system or buy a pre-installed one.

Input

External keyboard and mouse are required to provide input to the Raspberry Pi. No any additional software is needed to use the keyboard and mouse.

Monitor

We can use monitor or TV with HDMI port or DVI inputs as the screen for the Raspberry Pi. For DVI inputs, HDMI-to-DVI converters are required which can be finding easily in a market. Monitor is most important for the Raspberry Pi as it is the only way to see what we have done on it.

Network

As in laptop or computer, we can access to internet and network in Raspberry Pi as well. For that, we can use wired Ethernet connection which is easier option or Wi-Fi module to access Wi-Fi in the Raspberry Pi.

Power supply unit

A power supply (sometimes known as a power supply unit or PSU) is a device or system that supplies electrical or other types of energy to an output load or group of loads. The term is most commonly applied to electrical energy supplies, less often to mechanical ones, and rarely to others.

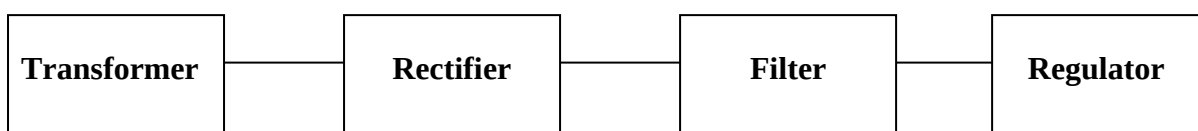


Fig4.2 Block diagram of power supply

The transform steps up or steps down the input line voltage and isolates the power supply from the power line. The RECTIFIER section converts the alternating current input signal to a pulsating direct current. However, as you proceed in this chapter you will learn that pulsating dc is not desirable. For this reason a FILTER section is used to convert pulsating dc to a purer, more desirable form of dc voltage.

The final section, the REGULATOR, does just what the name implies. It maintains the output of the power supply at a constant level in spite of large changes in load current or input line voltages. Now that you know what each section does, let's trace an ac signal through the power supply. At this point you need to see how this signal is altered within each section of the power supply. Later on in the chapter you will see how these changes take place. An input signal of 115 volts ac is applied to the primary of the transformer.

The transformer is a step-up transformer with a turns ratio of 1:3. You can calculate the output for this transformer by multiplying the input voltage by the ratio of turns in the primary to the ratio of turns in the secondary; therefore, $115 \text{ volts ac} \times 3 =$

345 volts ac (peak-to- peak) at the output. Because each diode in the rectifier section conducts for 180 degrees of the 360-degree input, the output of the rectifier will be one-half, or approximately 173 volts of pulsating dc. The filter section, a network of resistors, capacitors, or inductors, controls the rise and fall time of the varying signal; consequently, the signal remains at a more constant dc level. You will see the filter process more clearly in the discussion of the actual filter circuits. The output of the filter is a signal of 110 volts dc, with ac ripple riding on the dc. The reason for the lower voltage (average voltage) will be explained later in this chapter. The regulator maintains its output at a constant 110-volt dc level, which is used by the electronic equipment (more commonly called the load).

Simple 5V power supply for digital circuits

- Brief description of operation: Gives out well-regulated +5V output, output current capability of 100 mA
- Circuit protection: Built-in overheating protection shuts down output when regulator IC gets too hot
- Circuit complexity: Very simple and easy to build
- Circuit performance: Very stable +5V output voltage, reliable operation
- Availability of components: Easy to get, uses only very common basic components
- Design testing: Based on datasheet example circuit, I have used this circuit successfully as part of many electronics projects
- Applications: Part of electronics devices, small laboratory power supply
- Power supply voltage: Unregulated DC 8-18V power supply
- Power supply current: Needed output current + 5 mA
- Component costs: Few dollars for the electronics components + the input transformer cost

Image Processing on Raspberry Pi

Today image processing are used in various techniques, this paper presents the implementation of image processing operations on Raspberry Pi. The Raspberry Pi is a basic embedded system and being a low cost a single board computer used to

reduce the complexity of systems in real time applications. This platform is mainly based on python. Raspberry pi consist of Camera slot Interface (CSI) to interface the raspberry pi camera. Here, the Dark and Low contrast images captured by using the Raspberry Pi camera module are enhanced in order to identify the particular region of image. This concept is used in the real time application of MAV, The MAVs are basically used to capture images and videos through the Raspberry pi camera module. Because of its credit card sized (small) and less weight in the design. However, the image captured by MAVs will consist of unwanted things due to atmospheric conditions; hence it is necessary to remove noise present in the MAVs images.

INTRODUCTION

The image processing is a form of signal processing where the input is an image, photograph or video frame, the output of an image processing may be either an image or a video frame or a set of characteristics or parameters related to the image. The acquisition of digital image usually suffers from undesirable camera shakes and due to unstable random camera motions. Hence image enhancement algorithms are required to remove these unwanted camera shakes. This image processing concepts are implemented in Raspberry pi in the application of MAV. The Raspberry Pi is a basic embedded system having a credit card-sized single board computers developed in the UK by the Raspberry Pi Foundation. The Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SOC) which includes an ARM1176JZF-S Core (ARM V6K)700 MHz CPU processor, Broadcom Video Core IV GPU having 17 pins, 3.5W of power, and 512 MB of RAM memory. The Raspberry Pi system has Secure SD card reader (models A and B) or Micro SD card reader (models A+ and B+) sockets for boot media and persistent storage. The system provides Debian Linux operating system Raspbian image for download. Python is used as main programming language for raspberry pi. A micro air vehicle (MAV) is a remote-controlled, Unmanned Aircraft Vehicle (UAV) significantly smaller than typical UAVs that have a size restriction. UAV is an aircraft without a human pilot. Its flight is controlled either autonomously on board computers or by the remote control of a pilot on the ground or in another vehicle. By having a Raspberry Pi camera module available on a MAV the efficiency of this air vehicle increases and new fields of applications are available. It is needed in military Operations, in which targets have to be identified. Such identification is often done by

a human on ground, to reduce the probability of mistakes. But a Raspberry Pi camera module is also helpful if a MAV shall autonomously fly through an arch.

BASIC CONCEPT OF IMAGE PROCESSING

In general, any digital image processing algorithm consists of three stages: input, processor and output. In the input stage image is captured by a camera. It sent to a particular system to focus on a pixel of image that's gives, its output as a processed image.

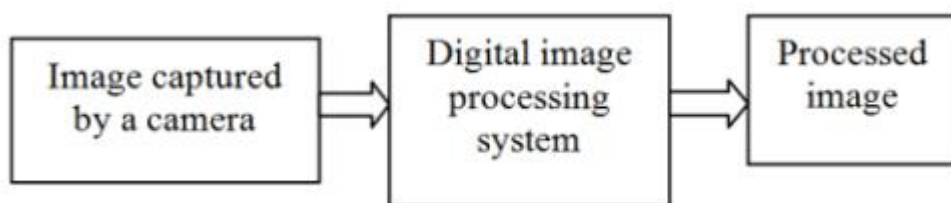


Fig 4.3 General Block diagram of image processing

SYSTEM HARDWARE DESIGN

The Raspberry Pi board is the central module of the whole embedded image capturing and processing system as given in fig. 2. Its main parts include: main processing chip unit, memory, power supply HDMI Out i.e VGA display, Ethernet port, and USB ports.

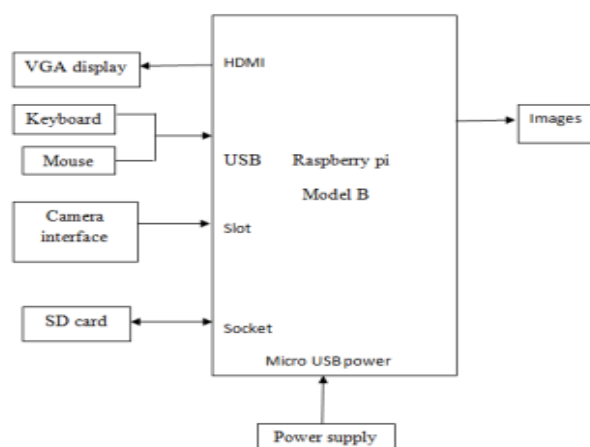


Fig 4.4 System Block Diagram

4.4 SOFTWARE DESCRIPTION

Start up with Raspberry Pi

Installation of operating system on Raspberry Pi

Raspberry Pi is a small computer; hence operating system (OS) should be installed. As the Raspberry doesn't have hard drive, OS is installed in the external memory. For that, memory card (SD card) is used for the installation of operating system and all the required software and supporting files are stored in the same SD card.

There are different types of operating system but we preferred to talk about NOOBS (New Out of the Box Software) as it is suitable for the beginners. We can either buy a preinstalled SD card or empty SD card. In pre-installed SD card, NOOBS is already copied and ready to boot. The detail description of installation of operating system in a blank SD card in step wise is given in Appendix B.

Use of laptop screen, keyboard and mouse for Raspberry Pi

If you don't have suitable screen for the Raspberry Pi, we can use the monitor of a laptop. Besides, we can use keyboard and mouse of laptop as well for Raspberry Pi. For that, two software are needed to install in a laptop which screen is going to use for Raspberry Pi. Installation and configuration of the software is discussed briefly in the following section. Finally, click "Next" button shown in Figure 12-3, it goes to last configuration dialogue box where "Finish" button is needed to click for the completion of the setting. After Completion of configuration, double click the application named as "Xming".

PuTTY

This is the primary software need to be installed. It can be downloaded in the provided link as follow, Download Putty. As, it is downloaded, it needs to be installed following some few normal steps of installation. For Configuration, double click the icon of Putty after the completion of installation and enter the IP address of Raspberry Pi as shown in Figure 12-4.



Fig 4.5 PuTTY login windows

In this window, username and password of the Raspberry Pi should be entered. The default username for a Raspberry Pi is pi. Press enter after entering correct username and password and you will see the windows where you should enter the text "lesson" as shown in Figure 12-7.

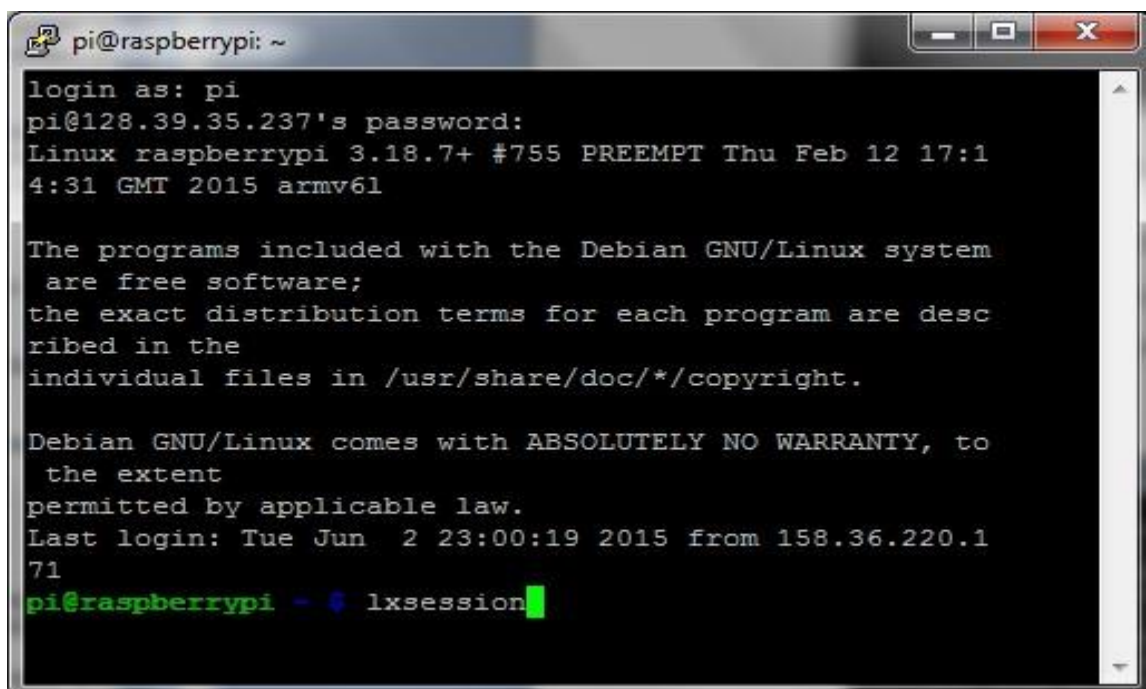


Fig 4.6 PuTTY success windows

Then, you will be able to see the desktop of Raspberry Pi in the laptop as shown in Figure 12-8 and mouse and keyboard can be used for the Raspberry Pi.

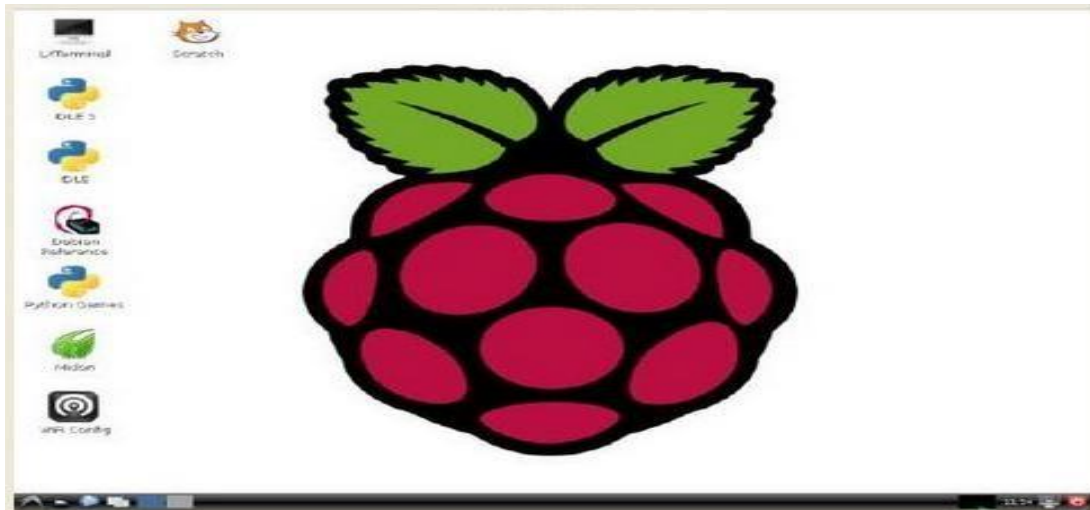


Figure 4.7 Raspberry Pi Desktop on your Laptop Display

Installation of required applications on Raspberry Pi

There are many applications that are needed to install in the Raspberry Pi for the completion of the thesis. For data logging, MySQL apache5 and php my admin are needed to install whereas for the web-page development, PHP is needed to install. Web page is used for the monitoring and managing purpose

Raspbian OS

Raspberry Pi OS (formerly Raspbian) is a Debian-based operating system for Raspberry Pi. Since 2015, it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the Raspberry Pi family of compact single-board computers. The first version of Raspbian was created by Mike Thompson and Peter Green as an independent project The initial released build was completed on July 15, 2012'

Raspberry Pi OS is highly optimized for the Raspberry Pi line of compact single-board computers with ARM CPUs. It runs on every Raspberry Pi except the Pico microcontroller. Raspberry Pi OS uses a modified LXDE as its desktop environment with the Openbox stacking window manager, along with a unique theme. The

distribution is shipped with a copy of the algebra program Wolfram Mathematica and a version of Minecraft called Minecraft: Pi Edition (note that Minecraft: Pi Edition is no longer installed as of the Debian bullseye update) as well as a lightweight version of the Chromium web browser.

User Interface

Raspberry Pi OS's desktop environment, PIXEL, looks similar to many common desktops, such as macOS, Microsoft Windows, and is based on LXDE. The menu bar is positioned at the top and contains an application menu and shortcuts to Terminal, Chromium, and File Manager. On the right is a Bluetooth menu, a Wi-Fi menu, volume control, and a digital clock.

Components

PCManFM is a file browser allowing quick access to all areas of the computer, and was redesigned in the first Raspberry Pi OS Buster release (2019-06-20).

Raspberry Pi OS originally used Epiphany as the web browser, but switched to Chromium with the launch of its redesigned desktop.

Raspberry Pi OS comes with many beginner IDEs, such as Thonny Python IDE, Mu Editor, and Greenfoot. It also ships with educational software like Scratch and Bookshelf.

Version Size

The Raspberry Pi documentation recommends at least a 4GB microSD card for Raspberry Pi OS Lite, an 8GB microSD card for Raspberry Pi OS, and a 16GB microSD card for Raspberry Pi OS Full.[19] The image files themselves are 1,875MB, 3,980MB, and 8,603MB respectively.

Python

What is Python?

Python: Dynamic programming language which supports several different programming paradigms:

- Procedural programming
- Object oriented programming

- Functional programming

Standard: Python byte code is executed in the Python interpreter (similar to Java) → platform independent code

Why Python?

Extremely versatile language Website development, data analysis, server maintenance, numerical analysis,

- Syntax is clear, easy to read and learn (almost pseudo code)
- Common language
- Intuitive object oriented programming
- Full modularity, hierarchical packages
- Comprehensive standard library for many tasks
- Big community Simply extendable via C/C++, wrapping of C/C++ libraries
- Focus: Programming speed

History

Start implementation in December 1989 by Guido van Rossum (CWI)

- 16.10.2000: Python 2.0
- Unicode support
- Garbage collector
- Development process more community oriented
- 3.12.2008: Python 3.0
- Not 100% backwards compatible 2007 & 2010 most popular programming language (TIOBE Index)
- Recommendation for scientific programming (Nature News, NPG, 2015)
- Current version: Python 3.9.2
- Python2 is out of support

Is Python fast enough?

- For user programs: Python is fast enough!
- Most parts of Python are written in C
- For compute intensive algorithms: Fortran, C, C++ might be better
- Performance-critical parts can be re-implemented in C/C++ if necessary
- First analyses, then optimize!

Conclusion

Python is a fully-fledged programming language. Any code you download from github, the Python Package Index, or anywhere else, can be malicious and would almost certainly go unnoticed by most people's firewalls.

OpenCV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. Let's start the chapter by defining the term "Computer Vision"

Computer Vision

Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modeling and replicating human vision using computer software and hardware.

Computer Vision overlaps significantly with the following fields:

- Image Processing: It focuses on image manipulation.
- Pattern Recognition: It explains various techniques to classify patterns.
- Photogrammetry: It is concerned with obtaining accurate measurements from images.

Computer Vision Vs Image Processing

Image processing deals with image-to-image transformation. The input and output of image processing are both images.

Computer vision is the construction of explicit, meaningful descriptions of physical objects from their image. The output of computer vision is a description or an interpretation of structures in 3D scene.

Applications of Computer Vision

Here we have listed down some of major domains where Computer Vision is heavily used. **Robotics Application**

- Localization — Determine robot location automatically
- Navigation
- Obstacles avoidance
- Assembly (peg-in-hole, welding, painting)
- Manipulation (e.g. PUMA robot manipulator)
- Human Robot Interaction (HRI): Intelligent robotics to interact with and serve people

Medicine Application

Classification and detection (e.g. lesion or cells classification and tumor detection)

- 2D/3D segmentation
- 3D human organ reconstruction (MRI or ultrasound)
- Vision-guided robotics surgery

Industrial Automation Application

- Industrial inspection (defect detection)
- Assembly
- Barcode and package label reading
- Object sorting

- Document understanding (e.g. OCR)

Security Application

- Biometrics (iris, finger print, face recognition)
- Surveillance — Detecting certain suspicious activities or behaviors Transportation

Application

- Autonomous vehicle
- Safety, e.g., driver vigilance monitoring

Features of OpenCV Library

Using OpenCV library, you can –

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it. OpenCV was originally developed in C++. In addition to it, Python and Java bindings were provided. OpenCV runs on various Operating Systems such as windows, Linux, OSx, FreeBSD, Net BSD, Open BSD, etc.

This tutorial explains the concepts of OpenCV with examples using Java bindings.

OpenCV Library Modules

Following are the main library modules of the OpenCV library.

Core Functionality

This module covers the basic data structures such as Scalar, Point, Range, etc., that are used to build OpenCV applications. In addition to these, it also includes the multidimensional array Mat, which is used to store the images. In the Java library of OpenCV, this module is included as a package with the name org.opencv.core.

Image Processing

This module covers various image processing operations such as image filtering, geometrical image transformations, color space conversion, histograms, etc. In the Java library of OpenCV, this module is included as a package with the name `org.opencv.imgproc`.

Video

This module covers the video analysis concepts such as motion estimation, background subtraction, and object tracking. In the Java library of OpenCV, this module is included as a package with the name `org.opencv.video`.

Video I/O

This module explains the video capturing and video codecs using OpenCV library. In the Java library of OpenCV, this module is included as a package with the name `org.opencv.videoio`.

calib3d

This module includes algorithms regarding basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence and elements of 3D reconstruction. In the Java library of OpenCV, this module is included as a package with the name `org.opencv.calib3d`.

features2d

This module includes the concepts of feature detection and description. In the Java library of OpenCV, this module is included as a package with the name `org.opencv.features2d`.

Objdetect

This module includes the detection of objects and instances of the predefined classes such as faces, eyes, mugs, people, cars, etc. In the Java library of OpenCV, this module is included as a package with the name `org.opencv.objdetect`.

Highgui

This is an easy-to-use interface with simple UI capabilities. In the Java library of OpenCV, the features of this module is included in two different packages namely, `org.opencv.imgcodecs` and `org.opencv.videoio`.

OpenCV — Storing Images

To capture an image, we use devices like cameras and scanners. These devices record numerical values of the image (Ex: pixel values). OpenCV is a library which processes the digital images, therefore we need to store these images for processing. The Mat class of OpenCV library is used to store the values of an image. It represents an n-dimensional array and is used to store image data of grayscale or color images, voxel volumes, vector fields, point clouds, tensors, histograms, etc. This class comprises of two data parts: the header and a pointer

- Header: Contains information like size, method used for storing, and the address of the matrix (constant in size).
- Pointer: Stores the pixel values of the image (Keeps on varying).

The Mat Class

The OpenCV Java library provides this class with the same name (Mat) within the package org.opencv.core.

Constructors

The Mat class of OpenCV Java library has various constructors, using which you can construct the Mat object.

1. This is the default constructor with no parameters in most cases. We use this to constructor to create an empty matrix and pass this to other OpenCV methods.
 2. Mat(int rows, int cols, int type) This constructor accepts three parameters of integer type representing the number of rows and columns in a 2D array and the type of the array (that is to be used to store data).
 3. Mat(int rows, int cols, int type, Scalar s) Including the parameters of the previous one, this constructor additionally accepts an object of the class Scalar as parameter.
 4. Mat(Size size, int type)
3. OpenCV — Storing Image

CHAPTER 5

RESULTS AND CONCLUSION

5.1 RESULT:

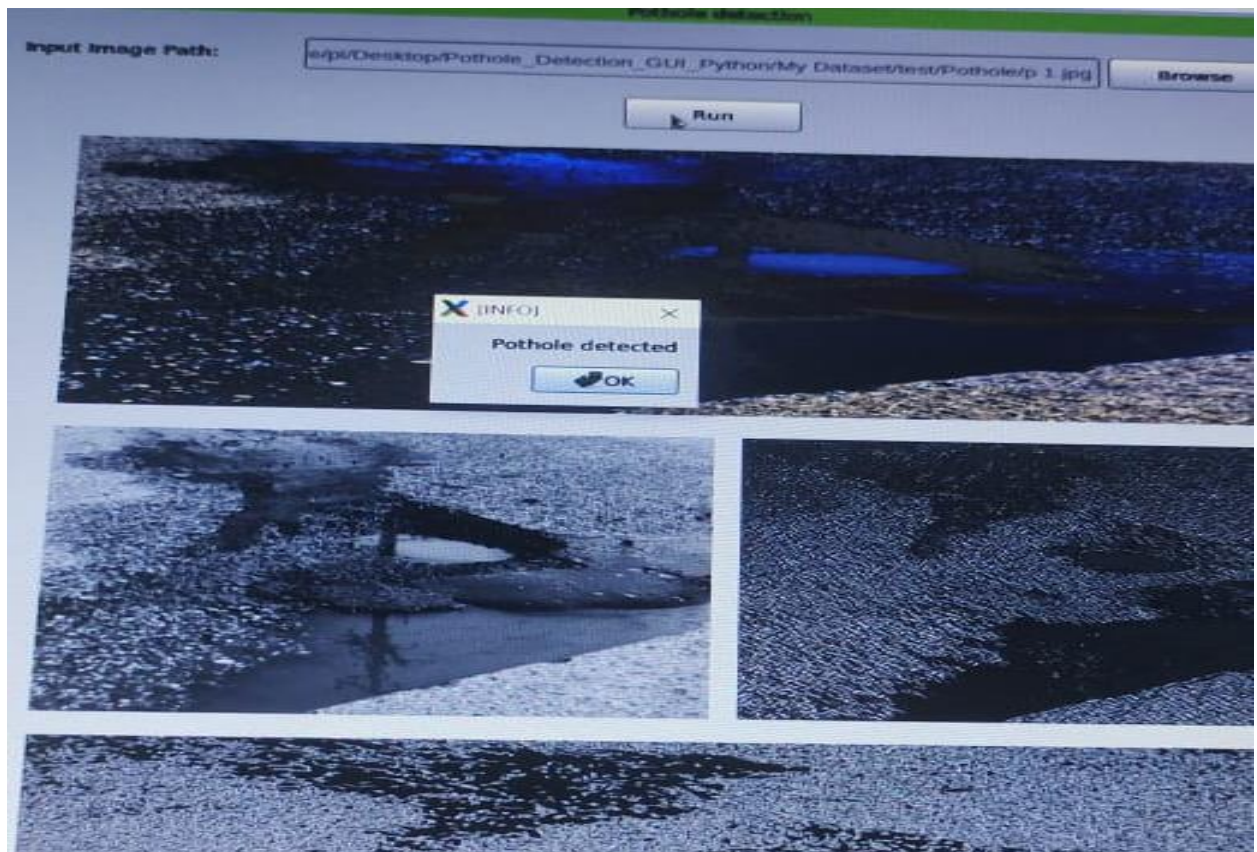


Fig 5.1 OUTPUT IMAGE

The reason for this output is abnormalities like potholes, uneven roads are on the images of the output, while analysing images with the images given inside the dataset, the output will be depicted as like this, through image analysis which have been done.



Fig 5.2 OUTPUT IMAGE

The reason for this output is abnormalities like potholes, uneven roads are not on the images of the output, while analysing images with the images given inside the dataset, the output will be depicted as like this, through image analysis which have been done.

CHAPTER 6

CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSION

In this paper, we have described a prototype system to read printed text on hand-held objects for assisting blind persons. In order to solve the common aiming problem for blind users, we have proposed a motion-based method to detect the object of interest, while the blind user simply shakes the object for a couple of seconds. This method can effectively distinguish the object of interest from background or other objects in the camera view. To extract text regions from complex backgrounds, we have proposed a novel text localization algorithm based on models of stroke orientation and edge distributions. The corresponding feature maps estimate the global structural feature of text at every pixel. Block patterns project the proposed feature maps of an image patch into a feature vector.

6.2 FUTURE SCOPE

Our future work will extend our localization algorithm to process text strings with characters fewer than three and to design more robust block patterns for text feature extraction. We will also extend our algorithm to handle no horizontal text strings. Furthermore, we will address the significant human interface issues associated with reading text by blind users. It has been developed by integrating features of all the hardware components and software used. In this paper, the camera acts as input for the paper. As the Raspberry Pi board is powered the camera starts streaming. The streaming data will be displayed on the screen using GUI application. When the object for label reading is placed in front of the camera then the capture button is clicked to provide image to the board. Using Tesseract library the image will be converted into data and the data detected from the image will be shown on the status bar. The obtained data will be pronounced through the ear phones using Flite library. Presence of every module has been reasoned out and placed carefully thus contributing to the best working of the unit. In this by using highly advanced ARM11 board this paper has been implemented.

APPENDIX

```
import sys from PyQt5.QtWidgets
import QApplication, QWidget, QLabel, QPushButton, QMessageBox, QLineEdit,
FileDialog
from PyQt5.QtGui import QFont, QIcon, QPixmap, QImage from PyQt5 import
QtCore
import time
import numpy as np
import cv2
import glob
# from tensorflow
import keras from tensorflow.keras
    import Sequential
    from tensorflow.keras.models
    import load_model from sklearn.preprocessing
    import LabelBinarizer from sklearn.model_selection
    import train_test_split from keras.utils
    import np_utils

import os
import cv2
#from PIL import Image
global size size = 100
def App():
if not QApplication.instance():
app = QApplication(sys.argv)
else:
app = QApplication.instance()
    Title = 'PyQt5'
    Init_UI = Q Widget()
```

```

Init_UI.resize(1000,700)
Init_UI.setWindowTitle(Title)
Init_UI.setGeometry(350, 100, 800, 1000)
title_label_color = QLabel(Init_UI)
title_label_color.setText("")
title_label_color.setGeometry(0,5,800,20)
title_label_color.setStyleSheet("background-color:lightgreen")
title_label_color.show()
title_label = QLabel(Init_UI)
title_label.setText('Pothole detection')
title_label.setGeometry(350, 5, 800, 20)
title_label.show()
input_img_label=QLabel(Init_UI)   input_img_label.setText('InputImagePath:')
input_img_label.setGeometry(15, 50, 150, 30)
#input_img_label.setStyleSheet("background-color:lightgreen")
input_img_label.show()
    img_textbox = QLineEdit(Init_UI)
    img_textbox.setGeometry(170,50,450,30)
    img_textbox.setText('Pleaseinputtheimage...')
    img_textbox.setFont(QFont("Arial",10))
    img_textbox.show()
    browse_btn = QPushButton(Init_UI)
    browse_btn.setText('Browse')
    browse_btn.setGeometry(625, 50, 100, 30)
    browse_btn.show(
browse_btn.clicked.connect(lambda: ) Select_Input_Img(img_textbox))

output_label_color = QLabel(Init_UI)
output_label_color.setText("")
output_label_color.setGeometry(0, 970, 800, 30)
output_label_color.setStyleSheet("background-color:lightgreen")
output_label_color.show()
output_label=QLabel(Init_UI)
output_label.setText("")

```

```

output_label.setGeometry(350,970,800,30)
output_label.show()
#80FAD579-69DB-4057-AC22-AED3AB66BDDD.gif
img1_label = QLabel(Init_UI)
img1_label.setPixmap(pixmap1)
img1_label.show()
img2_label = QLabel(Init_UI)\
pixmap2=QPixmap('80FAD579-69DB-4057-AC22AED3AB66BDDD.gif')
img2_label.setGeometry(QRect(50,420,355,25))
img2_label.setScaledContents(True)
img2_label.setPixmap(pixmap2)
img2_label.show()
img3_label = QLabel(Init_UI)
pixmap3=QPixmap('80FAD579-69DB-4057-AC22-AED3AB66BDDD.gif')
#img2_label.setGeometry(QRect(50, 420, 710, 250))
img3_label.setGeometry(QRect(420, 420, 355, 250))
img3_label.setScaledContents(True)
img3_label.setPixmap(pixmap3)
img3_label.show()
img4_label = QLabel(Init_UI)
pixmap4=QPixmap('80FAD579-69DB-4057-AC22-AED3AB66BDDD.gif')
img4_label.setGeometry(QRect(50,690,710,250))
img4_label.setScaledContents(True)
img4_label.setPixmap(pixmap4)
img4_label.show()
"status_label_color = QLabel(Init_UI)
status_label_color.setText("")
status_label_color.setGeometry(500,100,100,30)
status_label_color.setStyleSheet("background-color:yellow")
status_label_color.show()

status_label = QLabel(Init_UI)
status_label.setText('---Progress---')
status_label.setGeometry(500, 100, 100, 30)

```

```

status_label.show()""
detect_btn=QPushButton(Init_UI)
detect_btn.setText('Run')
detect_btn.setGeometry(350,100,100,30)
detect_btn.show()
detect_btn.clicked.connect(lambda:
Run_Predict(Init_UI,img_textbox,output_label_color,output_label,img_label,img2_l
abel,img3_label,img4_label))
Init_UI.show()
sys.exit(app.exec_())
print('Completed')
def
Run_Predict(Init_UI,img_textbox,output_label_color,output_label,img_label,img2_l
abel,img3_label,img4_label):
# Initialize
output_label_color.setStyleSheet("background-color:lightgreen")
output_label.setText("")
"status_label.setText(' Processing...')
status_label.show()""    print('Processing...')
os.system('espeak -v m3 -s 125 -a 125 "Processing"')
os.system('espeak -v m3 -s 125 -a 125 "Please wait"')

#80FAD579-69DB-4057-AC22-AED3AB66BDDD.gif

Input_Image_Path = img_textbox.text()
if ".jpg" not in Input_Image_Path:
mbox = QMessageBox()
mbox.setWindowTitle("Warning")
mbox.setGeometry(QtCore.QRect(600,350,200,100))        mbox.setText("Please
select a valid image and try again!")
mbox.setStandardButtons(QMessageBox.Ok)
mbox.exec_()
return img1_label = QLabel(Init_UI)
pixmap1=QPixmap('80FAD579-69DB-4057-AC22-AED3AB66BDDD.gif')

```

```

img1_label.setGeometry(QRect(50, 150, 710, 250))
img1_label.setScaledContents(True) img1_label.setPixmap(pixmap1)
img1_label.show()
#time.sleep(2)
img = cv2.imread(Input_Image_Path)
img_gray=cv2.cvtColor(img,
cv2.COLOR_BGR2GRAY)
cv2.imwrite('img_gray.jpg', img_gray)
img_blur=cv2.GaussianBlur(img_gray,(3,3),0)    cv2.imwrite('img_blur.jpg',
img_blur)
sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5) #
Sobel Edge Detection on the X axis
sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5) #
Sobel Edge Detection on the Y axis
sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5) #
Combined X and Y Sobel Edge Detection
cv2.imwrite('sobelx.jpg', sobelx)
cv2.imwrite('sobely.jpg', sobely)
cv2.imwrite('sobelxy.jpg', sobelxy)
edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200) # Canny
Edge Detection    cv2.imwrite('edges.jpg', edges)
#img1_label=QLabel(Init_UI)
pixmap1 = QPixmap(Input_Image_Path)
#img1_label.setGeometry(QRect(50, 150, 710, 250))
#img1_label.setScaledContents(True)    img1_label.setPixmap(pixmap1)
#img1_label.show()

#img2_label = QLabel(Init_UI) pixmap2 = QPixmap('img_blur.jpg')
#img2_label.setGeometry(QRect(50, 420, 355, 250))
#img2_label.setScaledContents(True) img2_label.setPixmap(pixmap2)
#img2_label.show()

#img3_label = QLabel(Init_UI) pixmap3 =

```

```

QPixmap('sobelxy.jpg')
#img3_label.setGeometry(QtCore.QRect(420, 420, 355, 250))
#img3_label.setScaledContents(True)
img3_label.setPixmap(pixmap3)
#img3_label.show()

#img4_label = QLabel(Init_UI)    pixmap4 = QPixmap('edges.jpg')
#img4_label.setGeometry(QtCore.QRect(50, 690, 710, 250))
#img4_label.setScaledContents(True)          img4_label.setPixmap(pixmap4)
#img4_label.show()
model = Sequential()
model=load_model('/home/pi/Desktop/Pothole_Detection_GUI_Python/sample1.h5')
potholeTestImages = glob.glob(Input_Image_Path)
for img in potholeTestImages:
    print(img)

test1 = [cv2.imread(img, 0) for img in potholeTestImages]
for i in range(0, len(test1)):
    test1[i] = cv2.resize(test1[i], (size, size))    temp3 = np.asarray(test1)
X_test = []
X_test.extend(temp3)
X_test = np.asarray(X_test)
X_test = X_test.reshape(X_test.shape[0], size, size, 1)
tests = model.predict_classes(X_test)
if tests[0] == 1:
    print('Potholedetected')    output_label_color.setStyleSheet("background-
color:red")    output_label.setText('Pothole detected')
    os.system('espeak -v m3 -s 125 -a 125 "Pothole Detected"') else:
    print('Pothole not detected')
    output_label_color.setStyleSheet("background-color: lightgreen")
    output_label.setText('Normal')

    os.system('espeak -v m3 -s 125 -a 125 "Normal"')

```

```

#status_label.setText(' Completed... ')
mbox = QMessageBox()
mbox.setWindowTitle("[INFO]")
mbox.setGeometry(QtCore.QRect(600,350,200,100))
mbox.setText("ProcessCompleted...")
mbox.setStandardButtons(QMessageBox.Ok)
mbox.exec_()
print('Completed')

def Select_Input_Img(img_textbox):
fileName, _ = QFileDialog.getOpenFileName(None,'Open file',
'/home/pi/Desktop/Pothole_Detection_GUI_Python/My Dataset',"Image files (*.jpg
*.png)")    if fileName:
img_textbox.setText(fileName)    else:        img_textbox.setText('Please input the
image.....')
if __name__ == "__main__":
App()

#end if

```

REFERENCES:

- [1] X. Chen and A. L. Yuille, "Detecting and reading text in natural scenes," in Proc. Comput. Vision Pattern Rec-ognit., 2004, vol. 2, pp. II-366–II-373.
- [2] S. Kumar, R. Gupta, N. Khanna, S. Chaudhury, and S. D. Joshi, "Text extraction and document image seg-mentation using matched wavelets and MRF model," IEEE Trans Image Process., vol. 16, no. 8, pp. 2117– 2128, Aug. 2007.
- [3] K. Kim, K. Jung, and J. Kim, "Texture-based ap-proach for text detection in images using support vec-tor machines and continuously adaptive mean shift al-gorith," IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, no. 12, pp. 1631–1639, Dec. 2003.
- [4] N. Giudice and G. Legge, "Blind navigation and the role of technology," in The Engineering Handbook of Smart Technology for Aging, Disability, and Indepen-dence, A. A. Helal, M. Mokhtari, and B. Abdulrazak, Eds. Hoboken, NJ, USA: Wiley, 2008.
- [5] World Health Organization. (2009). 10 facts about blindness and visual impairment.
- [6] Advance Data Reports from the National Health In-terview Survey (2008).
- [7] International Workshop on Camera-Based Docu-ment Analysis and Recognition (CBDAR 2005, 2007, 2009, 2011).
- [8] X. Chen, J. Yang, J. Zhang, and A. Waibel, "Auto-matic detection and recognition of signs from natural scenes," IEEE Trans. Image Process., vol. 13, no. 1, pp. 87–99, Jan. 2004.
- [9] D. Dakopoulos and N. G. Bourbakis, "Wearable ob-stacle avoidance electronic travel aids for blind: A sur-vey," IEEE Trans. Syst., Man, Cybern., vol. 40, no. 1, pp. 25–35, Jan. 2010.
- [10] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in Proc. Comput. Vision Pattern Recognit., 2010, pp. 2963–2970.

[11] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in Proc. Int. Conf. Machine Learning, 1996, pp. 148–156.

[13] An overview of the Tesseract OCR (optical character recognition) engine, and its possible enhancement for use in Wales in a pre-competitive research stage Prepared by the Language Technologies Unit (Canol-fan Bedwyr), Bangor University April 2008.

[14] A. Shahab, F. Shafait, and A. Dengel, "ICDAR 2011 robust reading competition: ICDAR Robust Reading Competition Challenge 2: Reading text in scene images," in Proc. Int. Conf. Document Anal. Recognit., 2011, pp. 1491–1496.

[15] KReader Mobile User Guide, knfb Reading Technology Inc. (2008).

[16] S. M. Lucas, "ICDAR 2005 text locating competition results," in Proc. Int. Conf. Document Anal. Recognit., 2005, vol. 1, pp. 80–84.

