SMART DRONE FOR PRECISION AGRICULTURE

a project report submitted by

GADDAM PRANEETH CHAND – (REG NO: UR16CS265)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

under the supervision of

Mrs. JEBAPRIYA, M.E., (Ph.D.)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed-to-be-University under Sec-3 of the UGC Act, 1956)
Karunya Nagar, Coimbatore - 641 114, India.

JUNE 2020

BONAFIDE CERTIFICATE

Certified that this project report **"SMART DRONE FOR PRECISION AGRICULTURE"** is the bonafide work of "AYYALA SAMUEL SUDEEP (UR16CS148)**"** who carried out the project work under my supervision.

S. Jeba Priya

SIGNATURE

Dr. J. Immanuel Johnraja

Head of the Department

Department of Computer Science and Engineering

SIGNATURE

Mrs. Jebapriya

Supervisor

Assistant Professor

Department of Computer Science and Engineering

Submitted for the Project Viva Voce held on

Examiner

DECLARATION

- I understand that Karunya Institute of Technology and Sciences Shall hold the **Copyrights of all thesis/dissertations** submitted to the University.
- I will republic the entire thesis/extracts of the thesis only with the permission of Karunya Institute of Technology and Sciences and I am liable to play 40% of royalty to Karunya Institute of Technology and Sciences.
- If I engage in documenting any research finding with an intention of publishing it for commercial purpose. **I shall obtain a NOC from the office of the registrar** prior to engaging in such activities.

UR16CS265

Signature of the Candidates with Registration Numbers

ACKNOWLEDGEMENT

First and foremost, we praise and thank **ALMIGHTY GOD** for giving us the will power and confidence to carry out our project.

We are grateful to our beloved founders Late **Dr. D.G.S. Dhinakaran**, **C.A.I.I.B, Ph.D.,** and **Dr. Paul Dhinakaran**, **M.B.A., Ph.D.,** for their love and always remembering us in their prayers.

We extend our thanks to **Dr. P. Mannar Jawahar, Ph.D.,** our honorable vice chancellor, **Dr. E. J. James, Ph.D., Dr. Ridling Margaret Waller, Ph.D.,** and **Dr. T. Lazar Mathew, Ph.D.,** our honorable Pro-Vice Chancellor(s) and **Dr. R. Elijah Blessing, Ph.D.,** our respected registrar for giving us the opportunity to carry out this project.

We are thankful to **Dr. G. Prince Arulraj, M.E., Ph.D.,** Dean (Engineering & Technology) for his support and encouragement.

We would like to place our heart-felt thanks and gratitude to **Dr. J. Immanuel Johnraja, Ph.D.,** HOD, Department of Computer Science and Engineering for his encouragement and guidance.

We are grateful to our guide, **Mrs. Jebapriya, M.E., (Ph.D.,)** Assistant Professor, Department of Computer Science and Engineering for his valuable support, advice and encouragement.

We also thank all the staff members of the Department for extending their helping hands to make this project work a success.

We would also like to thank all my friends and my parents who have prayed and helped me during the project work.

ABSTRACT

We propose developing a system that helps to find the disease of the Banana plant and find the particular pesticide that helps to cure the disease occurred using Image processing and Pattern matching techniques. Training of the model is done with the database of the healthy images and the diseased images contained different type of distinct classes of combinations. We have performed a survey of some research used deep learning techniques, applied on various images of the Banana plant. We have examined some particular agricultural problems in this study and all the metrics, sources, pre-processed data that has been used and worked under this. To perform the image processing and plant protection operations, an Unmanned Aerial Vehicle (UAV) based automatic control spraying system. UAVs are used to help the farmers in developing the best crops. As UAVs totally replace manned aircrafts and satellites. They have many advantages over some of the traditional remote-sensing techniques. UAVs can be easy to use in most of the drone mapping and data collection missions for conducting them autonomously which makes the drone to fly itself and capture data and process it to get the desired output which can be easier and cheaper to get. We have included the sudden spurt of UAV in some domains of agriculture. We outline some of the architectures and their usage in precision agriculture.

TABLE OF CONTENTS

TITLE	Page No:
BONAFIDE CERTIFICATE	**2**
DECLARATION	**3**
ACKNOWLEDGEMENT	**4**
ABSTRACT	**5**
TABLE OF CONTENTS	**6**
LIST OF FIGURES	**9**
LIST OF SYMBOLS AND ABBREVATIONS	**10**
LIST OF TABLES	**11**
1 INTRODUCTION	
1.1 Problem Overview	**12**
1.2 Objective of the project	
1.3 Problem Statement	
1.4 Chapter Wise Summary	**13**
2 SYSTEM ANALYSIS	
2.1 Existing System	**14**
2.2 Proposed system	
2.3 Use Case Analysis	
2.3.1 Use Case Diagram	**15**
2.4 Requirement Specifications	**16**
2.4.1 Functional Requirements	
2.4.2 Non Functional Requirements	
2.4.3 Hardware Requirements	
2.4.3.1 Brushless DC Motors	**17**
2.4.3.2 Propellers	

2.4.3.3 Aluminium Drone Frame
2.4.3.4 Electric Speed Controller 18
2.4.3.5 Transmitter and Reciever
2.4.3.6 Flight Controller 19
2.4.3.7 Battery
2.4.4 Software are Functional Requirements 20
2.4.4.1 Jupyter Notebook
2.4.4.2 Setting up Software Development Environment 22

3 SYSTEM DESIGN 24
3.1 Detailed Design
3.2 Description
3.3 Module Implementation 26
3.3.1 Imported Libraries 27
3.3.2 Conversion of RGB image into GRAY image
3.3.3 Threshold Application on the Image
3.3.4 Local Binary Pattern 28
3.4 Image Classification 30
3.5 Dataset Training 31
3.5.1 Imported Libraries
3.5.2 Initialising and Compiling CNN
3.5.3 Loading and Training the Dataset
3.6 Support Vector Machine (SVM) 32
3.7 Convolutional Neural Network

4 SYSTEM IMPLEMENTATION
4.1 Module Implementation 34
4.1.1 Introduction
4.1.2 Assembling Components
4.1.3 Bill of Materials
4.2 Implemented Modules for Pattern Technique 35
4.2.1 Modules and Software Used

4.3 Materials and methods **37**

4.3.1 METHODS and MATERIALS IN SOFTWARE

4.3.1.1 Dataset

4.3.1.2 Image Pre-processing and Labelling.

4.3.1.3 Classification of the Images.

4.4 Test Cases and Results **38**

4.5 Research Objectives

5 CONCLUSION **41**

Appendix A **42**

Appendix B **48**

REFERENCES **49**

LIST OF FIGURES

2.1	**Use-case diagram**	**15**
2.2	**AX_2810Q 900Kv motors**	**17**
2.3	**Propellers 12x4.5 inches**	**17**
2.4	**Aluminium hexa Drone frame**	**18**
2.5	**Skywalker 40A esc**	**19**
2.6	**FLYSKY FS16 Transmitter & Receiver**	**19**
2.7	**DJI NAZA M LITE with GPS COMPASS**	**20**
2.8	**Gens Ace 5500mAh Li-po Battery**	**20**
2.9	**DRONE**	**21**
3.1	**Spraying control system of a drone.**	**24**
3.2	**UAV Flight Control System chart**	**25**
3.3	**UAV structure and Functionality**	**25**
3.4	**LBP image of healthy, Black Sigatoka and Cordana leaf spot.**	**30**
3.5	**Convolutional neural network model**	**32**
4.1	**Block diagram of Disease detection system**	**36**
4.2	**LOCAL BINARY PATTERN OPERATION**	**36**
4.3	**Sample picture of** **(a) healthy banana leaf** **(b) Black Sigatoka disease** **(c) Cordana leaf spot**	**37**
4.4	**Accuracy of the trained dataset**	**38**
4.5	**Prediction of 50 images of healthy and not healthy plants**	**39**
4.6	**black sigatoka image tested**	**40**
4.7	**healthy image tested**	**40**

LIST OF ABBREVATIONS AND SYMBOLS

UAV	-	**Unmanned Aerial Vehicle**
CNN	-	**Convolutional Neural Network**
WSN	-	**Wireless Sensor Network**
LBP	-	**Local Binary Pattern**
SVM	-	**Support Vector Machine**
ESC	-	**Electronic Speed Control**

LIST OF TABLES

4.1 BILL OF MATERIALS 35

CHAPTER 1
Introduction

1.1. Project Overview

In this project, we propose developing a system that helps to find the disease of the Banana plant and find the particular pesticide that helps to cure the disease occurred using Image processing and Pattern matching techniques. Training of the model is done with the database of the healthy images and the diseased images contained different type of distinct classes of combinations. We have performed a survey of some research used deep learning techniques, applied on various images of the Banana plant. We have examined some particular agricultural problems in this study and all the metrics, sources, pre-processed data that has been used and worked under this. To perform the image processing and plant protection operations, an Unmanned Aerial Vehicle (UAV) based automatic control spraying system. UAVs are used to help the farmers in developing the best crops. As UAVs totally replace manned aircrafts and satellites. They have many advantages over some of the traditional remote-sensing techniques. UAVs can be easy to use in most of the drone mapping and data collection missions for conducting them autonomously which makes the drone to fly itself and capture data and process it to get the desired output which can be easier and cheaper to get. We have included the sudden spurt of UAV in some domains of agriculture. We outline some of the architectures and their usage in precision agriculture.

1.2. Objective of the project

- The main objective of this system is to find the plants that are effected by diseases using Image Classification.
- This system provides the classifier that filters out the image to give the output as the image that has been taken in the field is effected with the disease or not and informs the users about the disease.
- For Instance, Authors propose the image processing algorithm to classify the images that can be potentially used by the adversaries to construct the user's actual movements.

Drones and Urban farming is now equipped with good navigation systems and image processing systems that aids the farmers on getting the most accurate view on the present situation of the farm that they can be able to take the preventive measure to keep the plants safe.

1.3 PROBLEM STATEMENT

Some of the problems that are faced using traditional farming methods are:

- Usage of higher man power for the detection of the disease and prevention of it using pesticides.
- No service has been done to recognise the occurrence of the disease.

1.4 CHAPTER WISE SUMMARY

Chapter 2 – deals with the System Analysis where problems related to existing system will be mentioned. The main cause which led to the development of the proposed system. A use-case analysis of the scenario is shown. Some requirements such as functional requirements,
Non- functional requirements. Hardware requirements, Software requirements has been cited.

Chapter 3 – deals with the System design where a brief description of the architecture diagram is explained. The design of the methodology i.e., the system of methods we are using. Explanation of Modules followed by the database design.

Chapter 4 – deals with the System Implementation for all the modules and all testing is done in this phase.

Chapter 5 – deals with the conclusion i.e., what we can draw from this project and future scope of the project.

CHAPTER 2

SYSTEM ANALYISIS

The detection of plant leaf illnesses requires shut observation. Image processing strategies are employed to decorate the fine of images. Besides, the proposed work uses deep learning techniques and image processing to classify the Banana plant ailments which are cited above. The proposed gadget entails various steps, which include- dataset creation, image pre-processing and deep learning based training and classification. Finally, a device is developed that extracts the features from pictures and classifies the ailments is the usage of deep learning Algorithms.

2.1 Existing System

In the existing system, the image classification is done using the pictures taken by the camera manually. So there is more amount of man power involved in it. We have drones which can fly and spray through the field with water or pesticide which can be operated from some distance.If they are merged together that makes the disease identification easier with the use of the drones fit with the camera. However, this provides a tedious task to update the drone with image classification and pattern techniques.

2.2 Proposed System

This system presents a classification model that classifies diseased plants and healthy plants from where the data is taken from the camera that is fit with the drone. The image that is taken using the system we process the image to apply binary pattern techniques and filter the image using classifiers to detect the disease occurred to the plant. These kind of diseases can be found out using some of the Image processing techniques and classifying data using deep learning algorithms. The steps that are done for processing the data will be:

- Image Acquisition
- Image pre-processing
- Image feature Extraction
- Classification.

2.3 USE CASE ANALYSIS

2.3.1 USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system that shows relationship between the user and the different use case in which the user is involved.According to 2.3, the actors and the use case details are given in the Fig 2.1

The various use cases are:

- Remote control
- Drone camera
- Image classifier
- Interface for Drone and system

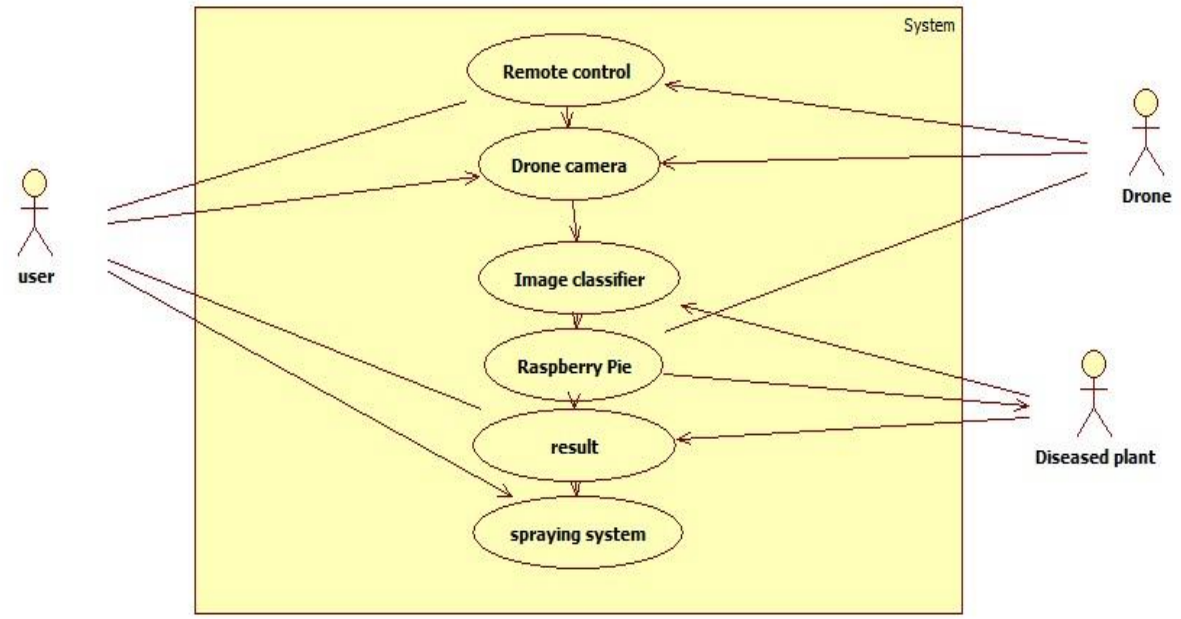


Fig 2.1 Use-case diagram

2.4 REQUIREMENT SPECIFICATIONS

Requirements specification is a complete description of the behaviour of the prototype to be developed. Requirements are categorised in several ways. This section covers functional and non-functional requirements of the prototype.

2.4.1 FUNCTIONAL REQUIREMENTS

Functional requirements explain what has done by the prototype by the identifying the necessary task, action or activity that must be accomplished.

They include:

- The prototype should be able to take the images from the camera fixed with the camera.
- The prototype should be able to convert given image into Local Binary Pattern which has the pixels with the binary values combined with a particular set of values.
- The prototype should be able to classify the image as the healthy or diseased using the classification technique.
- The prototype should inform about the occurrence of the disease to the banana plant.

2.4.2 NONFUNCTIONAL REQUIREMENTS:

Non-functional requirements are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviours. The key non-functional requirements identified in the prototype are security, performance, availability of image and reliability.

2.4.3 HARDWARE REQUIREMENTS

Rapid prototyping methodology is used in prototype development. Each unit of the drone was assembled ad tested before the entire prototype was assembled. It was important to ensure that each part of the drone was functioning independently and was able to take the data from the environment.

To develop and test the system, the following materials from the bill of materials was used:

2.4.3.1 BRUSHLESS DC MOTORS

Fig 2.2 AX_2810Q 900Kv motors

A Brush Less DC motor is also known as electrically commutated motor and synchronous DC motor. They are powered by direct current via an inverter or any power supply as AC to work through each phase of the motor using a closed loop controller. These help propellers to rotate on the drone. In this project we have used 900kv motors in the quantity of 6 for each propeller.

2.4.3.2 PROPELLERS

Fig 2.3 Propellers 12x4.5 inches

A propeller is a rotating hub with radiating blades that are set to a pitch to form a spiral, that transforms rotational power into linear thrust by acting upon linear thrust by acting upon water or air.

The rotational motion of the blades is converted into thrust by creating the pressure difference between the surfaces. In this project we have used 6piece propeller set to make the pressure difference and make the drone lift up.

2.4.3.3 ALUMINIUM DRONE FRAME:

The use of multiple Frame concept is for simplicity and flexibility, designed for the serious flyer and ideal for aerial photography. These frames are made of Carbon fibre, with custom moulded alloy arms, they are the strongest, stiffest and vibration free airframes you can find and considerably lighter than any other frame used.

2.4 Aluminium hexa Drone frame

2.4.3.4 ELECTRIC SPEED CONTROLLER:

The electric Speed Controller (ESC) is an electronic circuit that controls and regulates the speed of an electric motor. It can also provide reversing of the motor and dynamic braking. These are used in electrically powered radio controlled models. It follows a speed reference signal and varies the switching rate of a network. For this project we have used **Skywalker 40A esc** with radio controlled airplane.

Fig 2.5 Skywalker 40A esc

2.4.3.5 TRANSMITTER AND RECIEVER

A receiver is the opposite of the radio transmitter. It uses an antenna to capture radio waves, processes those waves to extract only those waves that are vibrating at a particular frequency. These are used to control signals transmitted by radio to remotely control a device. In this project we have used FLYSKY FS16 Transmitter & Receiver.

Fig 2.6 FLYSKY FS16 Transmitter & Receiver

2.4.3.6 FLIGHT CONTROLLER

A Flight controller is a small circuit board of varieties of complexity. Its function is to direct the rotations per second of each motor in the response given as input. A command from the pilot is fed into the flight controller which shows how to manipulate the motors accordingly. In this project we have used DJI NAZA M LITE with GPS COMPASS which is a multi-rotor flight controller has GPS device connected to plot the points where the diseased plant is found.

Fig 2.7 DJI NAZA M LITE with GPS COMPASS

2.4.3.7 BATTERY:

Gens ace 5500mAh 3S 11.1V 45C Lipo Battery Pack with Deans Plug

Fig 2.8 Gens Ace 5500mAh Li-po Battery

In our project we have used 5500mAh battery which gives us 15 minutes of Standby time for the drone which has connectors (deans plug) to connect the LiPo battery to the drone. These plugs have low resistance meaning longer run times and cooler batteries and gives better performance.

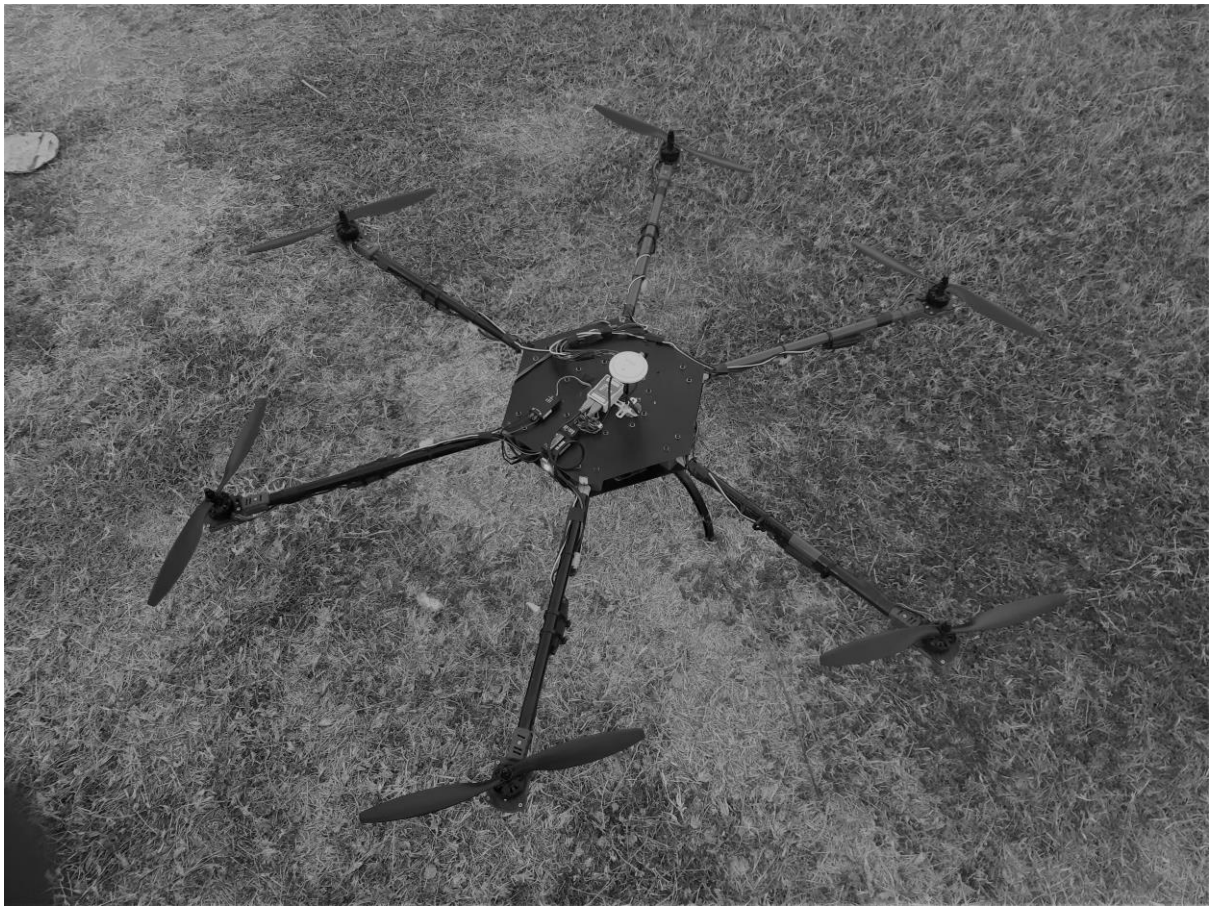


Fig 2.9 Drone

2.4.4 SOFTWARE FUNCTIONAL REQUIREMENTS

2.4.4.1 JUPYTER NOTEBOOK

According to its developers, formerly known as ipython notebooks created to develop open source software, open standards and services on an environment by supporting the programming languages which are Julia, python and R. Project Jupyter has developed and supported the interactive computing products like Jupyter notebook, JupyterHub and JupyterLab.

Jupyter Notebook (earlier IPython Notebooks) is an online intelligent computational condition for making Jupyter note book records. The "notebook" term can conversationally make reference to a wide range of elements, essentially the Jupyter web application, Jupyter Python web server, or Jupyter report design contingent upon setting. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries:

- IPython
- ØMQ
- Tornado (web server)
- jQuery
- Bootstrap (front-end framework)
- MathJax

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

2.4.4.2 SETTING UP SOFTWARE DEVELOPMENT ENVIROMENT

NumPy is the fundamental package for scientific computing with Python. It contains among other things: a powerful N-dimensional array object.

OpenCV (*Open source computer vision*) is a library of programming functions mainly aimed at real-time computer vision.

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the **Python** programming language.

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms

These are the libraries which are imported for image segmentation and classifying the image with the help of open source computer vision method and matplot library which is used to print the out of the image as a graph.

Keras is an open-source neural-network library written in Python. In our project this is working on TensorFlow in the backend. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

CHAPTER 3

SYSTEM DESIGN

3.1 DETAILED DESIGN

ARCHITECTURE DIAGRAMS

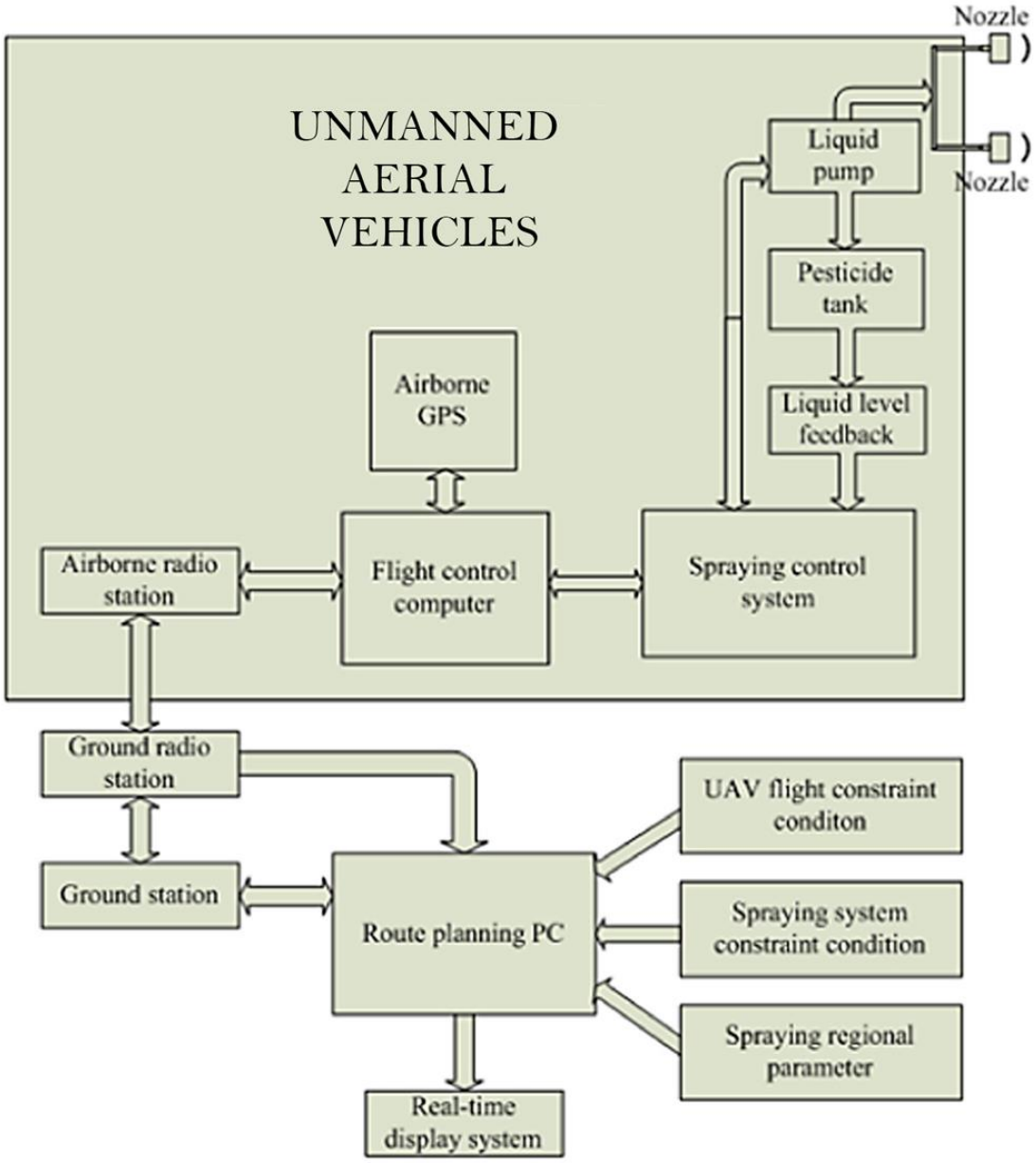


Fig 3.1 Spraying control system of a drone.

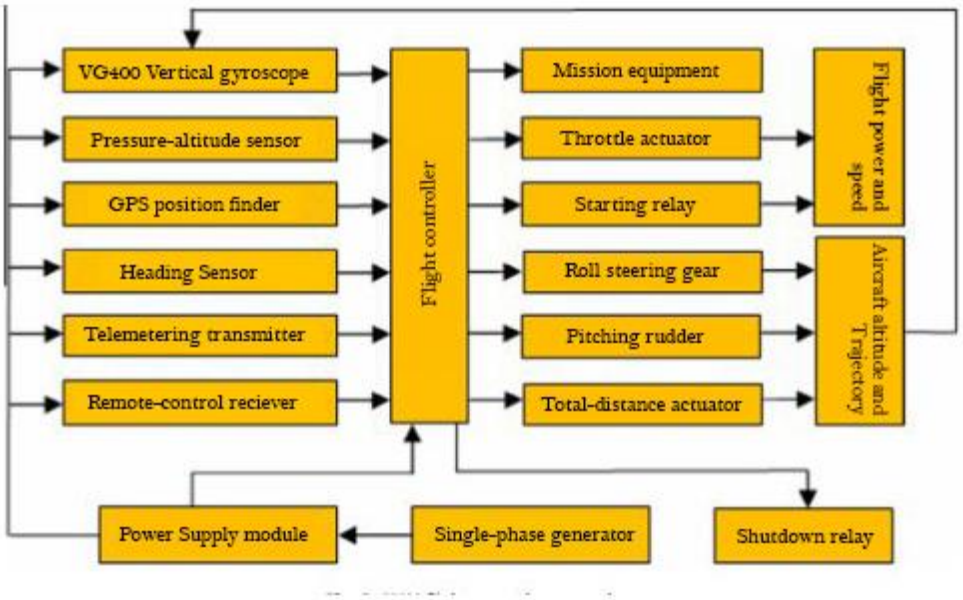


Fig 3.2: UAV Flight control System chart

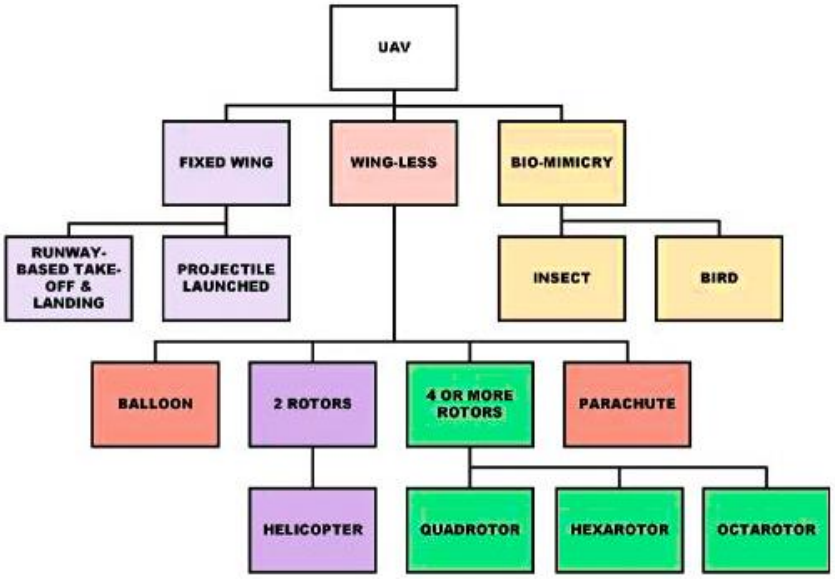


Fig 3.3 UAV structure and Functionality

3.2 DESCRIPTION:

The architecture comprises of all the core components that are used to construct a drone are discussed in the above diagrams. The brushless DC motors, battery, propellers, ESC, Hexa Aluminium frame ,GPS module are all connected to cumulatively constitute the physical element of the system .

We have divided our project into four phases such as phase1, phase2, phase3, phase 4.

In phase1 we have converted the given RGB image into local binary pattern which uses pattern technique to make the image as a neighbouring pixel intake that we define in python. This pattern technique uses matplotlib for the conversion process. Any grayscale image can be viewed as a topographic surface where high intensity denotes peaks and hills while low intensity denotes valleys. In the process of image pre-processing, instead *HSI* colour space model is used to provide luminance information of the plants. *H* denotes hue, *S* denotes saturation and *I* denotes intensity.

In phase2 we take the image from the drone and send it to our system to pre-process the picture using the source code in python it works in this way after compiling the code in python if there are no errors the picture will be resized and reshaped with the dimensions data set is loaded with.

In phase3, this is the phase where the data set is trained using CNN. We have applied Conv2D, maxPooling, Flatten to initialize the CNN. And the address of the data set storage is given and model is trained with 100 healthy images and 100 non healthy images. And the accuracy of the trained data set will be printed.

In phase4, this is the phase where we classify the image with the source code written in jupyter note book which will use binary classification to predict the image and show how much it is matching with the dataset and gives the name of the disease as the final output of the project.

3.3 MODULE IMPLEMENTATION

In the classification process, **Local Binary Pattern** algorithm is used to get the detail of the image with help of **Opencv.**

3.3.1 Imported Libraries

```
import cv2 as cv
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from PIL import Image
```

3.3.2 Conversion of RGB image into Gray image

```
pic = cv.cvtColor( pic , cv.COLOR_BGR2GRAY )

#displaying the grayscale picture

#cv.imshow( 'gray_scale' , pic )

cv.imshow( "pre-processed picture" , pic )
print(pic)
```

Any grayscale image can be viewed as a topographic surface where high intensity denotes peaks and hills while low intensity denotes valleys. In the process of image pre-processing, instead *HSI* colour space model is used to provide luminance information of the plants. *H* denotes hue, *S* denotes saturation and *I* denotes intensity.

3.3.3 Threshold application on the Image

```
def get_pixel(img, center, x, y):
    new_value = 0
    try:
        if img[x][y] >= center:
            new_value = 1
    except:
        pass
    return new_value
```

Thresholding point is the point at which we characterize the pixels that are esteemed in a picture. In OpenCV thresholding is done on grayscale pictures, which have pixel esteems running from 0–255. At the point when you limit a picture you arrange these pixels into bunches setting an upper and lower bound to each gathering. The threshold will be calculated as follows:

1. Select an initial estimate for T (using histogram or mean value – use ‘imhist’ or ‘mean2’ command) .

2. Segment the image using T. This will produce two groups of pixels: G_1 consisting of all pixels with gray level values > T and G_2 consisting of pixels with gray level values t T (e.g: g = f>T;)
3. Compute the average gray level values t_1 and t_2 for the pixels in regions G_1 (say g) and G_2 (say ~g)
4. Compute a new threshold value
5. T = 0.5 (t_1 + t_2)
6. Repeat steps 2 through 4 until the absolute difference in T in successive iterations is smaller than a predefined parameter T_o (say 0.5).

Additionally, in LBP, we calculate the thresholding values directly in binary format [0,1].
Where the G_1 is 1 and G_2 is 0.

3.3.4 LOCAL BINARY PATTERN (LBP)

Local Binary Pattern (LBP) is an efficient and uniform texture pattern technique used to describe the local texture patterns of an image. It is broadly exploited in many applications and it is based on local patterns instead of explaining it on pixel level. In LBP operation, the images are transformed into a matrix of integer values to describe the fine structure of an image at local level. For a 3*3 matrix, the eight neighborhood pixels are threshold with the center pixel. If the neighboring pixel is greater than the center pixel the neighbor pixel is set as 1 or otherwise zero. The threshold value is encoded into a decimal value called as
LBP code using the Eq (1). Mathematical expression of LBP is given as,

$$\boldsymbol{LBP_{P,R}}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c)^2 \quad (1)$$

where, g_c is the center pixel value, g_p is the neighborhood pixel value, P represents the number of neighborhood pixels and R represents the radius of the neighborhood

```
def LBP(pic,h,w):
    img = pic

    center = img[w][h]
    val_ar = []
    val_ar.append(get_pixel(img, center, w-1, h+1))      # top_right
    val_ar.append(get_pixel(img, center, w, h+1))        # right
    val_ar.append(get_pixel(img, center, w+1, h+1))      # bottom_right
    val_ar.append(get_pixel(img, center, w+1, h))        # bottom
    val_ar.append(get_pixel(img, center, w+1, h-1))      # bottom_left
    val_ar.append(get_pixel(img, center, w, h-1))        # left
    val_ar.append(get_pixel(img, center, w-1, h-1))      # top_left
    val_ar.append(get_pixel(img, center, w-1, h))        # top

    power_val = [1, 2, 4, 8, 16, 32, 64, 128]
    val = 0
    for k in range(len(val_ar)):
        val += val_ar[k] * power_val[k]
    return val
shape= np.shape(pic)
#print(shape)
height =shape[1]
width = shape[0]
img_lbp = np.zeros((height, width,3), np.uint8)
for i in range(0, height):
        for j in range(0, width):
            img_lbp[i, j] = LBP(pic, i, j)
im=Image.fromarray(img_lbp)
im.save('lbp.jpg')
cv.imshow("lbp",img_lbp)
#rotating
lbp = Image.open("lbp.jpg")
rotated_90 = lbp.transpose(Image.FLIP_LEFT_RIGHT)
rotated_90 = rotated_90.transpose(Image.ROTATE_90)
print(np.shape(rotated_90))
rotated_90.show()
```

and function s(t) is defined as:

$$s(t) = \begin{pmatrix} 1 & t \geq 0 \\ 0 & t < 0 \end{pmatrix}$$

The histogram features of size 2^P are extracted from the obtained LBP code of each pixel using the given expression:

$$H(k) = \sum_{n=0}^{I} \sum_{j=1}^{J} f\big(LBP_{N,R}(i,j), k\big), k \in [0, K]$$

$$f(x,y) = \begin{cases} 1, & x = y \\ 0, otherwise \end{cases}$$

where, *K* is the maximal LBP code value.

In this experiment the value of '*N*' and '*R*' are set to '8' and '1' respectively to compute the LBP feature. Hence, histogram feature vector length is 256 for 8 neighboring pixels obtained. The generalized process for computing LBP descriptor is presented in Fig 4.2. The LBP image of different banana leaf conditions are shown in Fig 6.1.

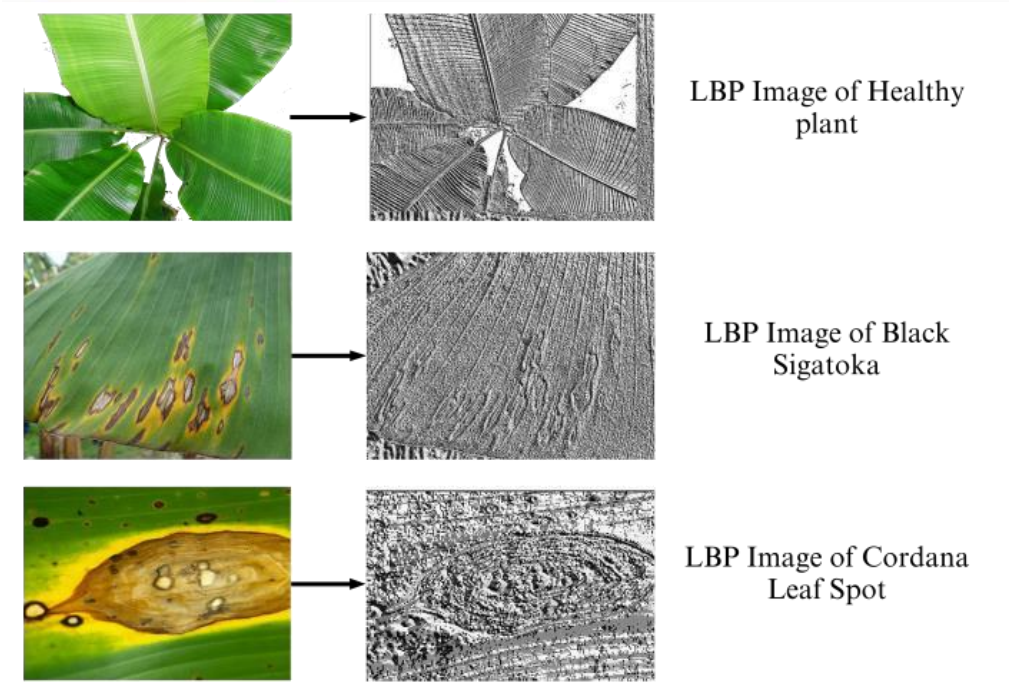


Fig 3.4 LBP image of healthy, Black Sigatoka and Cordana leaf spot.

To analyze the performance of the proposed algorithms SVM and CNN classifiers with 7-fold cross validation is used in this work. The statistical parameters like accuracy, sensitivity and specificity are used to examine the performance of the classifiers.

$$\text{Accuracy (Acc)} = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

$$\text{Senstivity (Sen)} = \left(\frac{TP}{TP+FN}\right)$$

$$\text{Specificity (Spe)} = \left(\frac{TN}{TN+FP}\right)$$

3.4 IMAGE CLASSIFICATION:

The model has been loaded from the json file. We are using the binary cross entropy to calculate accuracy of the model and the loaded images. Training model has been trained with 100

images of black sigatoka and the test set is having set of images mixed with diseased and healthy plants.

In the classifier we are classifying the model and checking the accuracy matching with the images loaded in the dataset.

The path for the images is given and the images are classified according to the given input and gives the value of how much does it match. It prints out the values of the number of sigatoka and healthy images is also predicted and that value is computed to get the value of accuracy.

3.5 DATASET TRAINING

3.5.1 Imported Libraries

Sequential is used to initialise the CNN to train the dataset. **Conv2D** is used to convert 2D image into the shape that we want to be for the input image. **MaxPooling2D** is used to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. **Flatten** is used to transform a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier.

3.5.2 Initialising and Compiling CNN

The process of classification is done using CNN sequential library initialises the neural network. It trains the dataset using the address given in the program. This compiles by applying layers on the dataset like Converting 2D image and pooling the image to get the parameters that can be used for the computational process. Then Flattening of the image will connect the matrix to the image.

3.5.3 Loading and Training the dataset

We use **Image Generator** to get the images and train the dataset from the directory by giving the source address of the dataset and test the dataset by using the test_datagen by the directory address.

3.6 Support Vector Machine (SVM)

Support Vector Machine (SVM) is one of the well-known strategies utilized for both linear and non-linear classification. Classification performed by SVM are more exact than other algorithms. It is utilized in numerous applications, particularly those including exceptionally high dimensional information. The higher performance of SVM classifier makes it a favored alternative for applications such as face recognition and plant disease detection. SVM utilizes structural risk minimization (SRM) principle to maximizes the margin of class separation for better

generalization performance of SVM. The input data is classified into two distinct classes by standard SVM classifier. Therefore, the optimal combination of the LBP descriptors and SVM classification can result in high plant disease discrimination accuracy. Hence, in this work SVM with cubic kernel function is employed to classify the healthy and diseased leaves. In SVM, the general solution used for finding the hyperplane with kernel functions are given as,

$$f(x) = \sum a_i y_i K(x_i, x) \quad (8)$$

where, $\{x_i, y_i\}$ is the training data with classes y_i, belonging to $\{-1, 1\}$, K is the kernel function.

3.7 CONVOLUTIONAL NEURAL NETWORK:

Convolutional Neural Networks are category of neural networks which has multiple convolutional layers and mostly used for image processing, classification and segmentation. It will be used to classify and filter over the input.

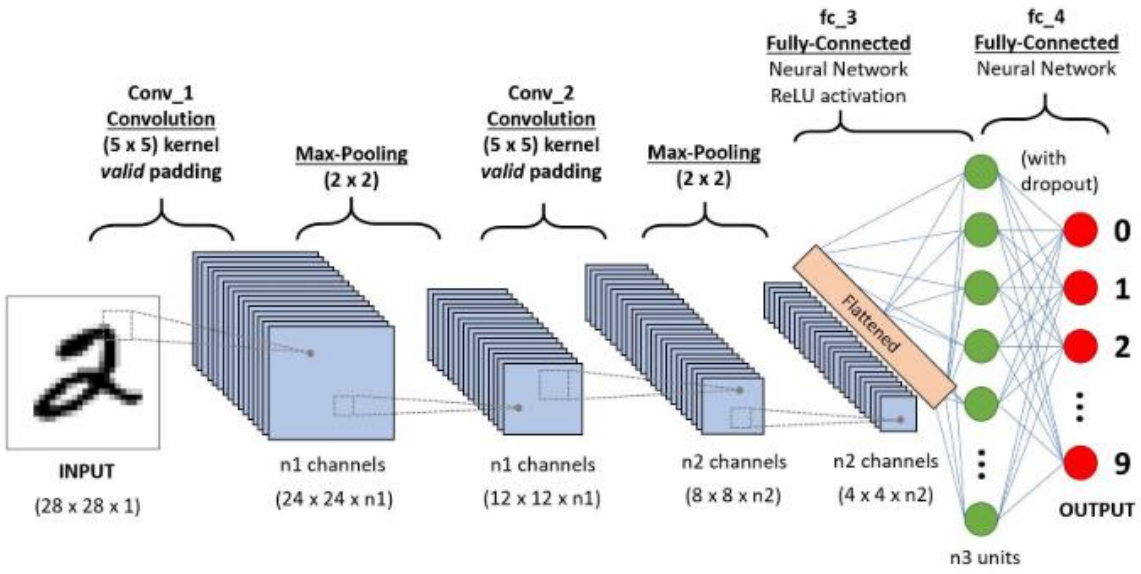


Fig 3.5 Convolutional neural network model

In our project the classifier program will work through layers applied the filtering of the output that has to be classified will be done by doing the comparison of the input image and prints the output.

We have trained the model with 100 images of healthy plant and 100 images of sigatoka effected plants. The testing is done to say it is effected then the output acquired from it is shown.

Our classification model has been tested with 78 images are combination of healthy and black sigatoka effected. So we have applied the whole folder of the 78 images to be predicted and the output is shown.

CHAPTER 4
SYSTEM IMPLEMENTATION

4.1 MODULE IMPLEMENATION

4.1.1 INTRODUCTION

This section covers implementation and the testing phase of the prototype this section also covers conclusions of the test results and the challenges faced during the implementation process. The various devices, software development platforms as well as test results has been discussed in this chapter. Rapid prototyping methodology discussed in chapter3 was fully implemented in the development of the prototype.

4.1.2 ASSEMBLING COMPONENTS

It was important to first assemble and configure all hardware and software components to ease the implementation and testing cycle. Assembling all components allows for adequate planning and resource allocation.

4.1.3 BILL OF MATERIALS

A bill of materials is a detailed list of items parts required for the development of the drone prototype. The bill of materials in table 4.1 below outlines a summary of hardware components used in the development of the prototype and technical specifications such as power supply are important because they directly affect the drone architecture despite their being number of manufacturers in the drone hardware business, we chose the devices listed in table4.1because they were affordable at the same time provided sufficient accuracy.

Material Name	Specifications	Description
AX_2810Q 900Kv motors	Power supply: 900Kv DC	It uses power supply to rotate the motors and make help the propellers to rotate.
Skywalker 40A esc	Current: 60A	It controls the speed of the drone with dynamic braking
DJI NAZA M LITE with GPS COMPASS COMBO	Flight controller with GPS tracker	Flight controller which can locate the position of the Drone
FLYSKY FS16 Transmitter & Receiver	Radio signals transmitted at range of 2.4GHz	Uses radio signals to control the moments of drone
Aluminium hexa frame	Dimensions:900mm	These are used to hold the propellers and motors
Gens Ace Li-po Battery	Capacity 5500mAh	Gives power supply
Propellers	Dimensions:12*4.5inch	They rotate to provide the pressure to raise drone

Table 4.1 Bill of materials

4.2 IMPLEMENTED MODULES FOR PATTERN TECHNIQUE

Local Binary Pattern (LBP) is a straightforward but still very productive Image processing pattern which marks the pixels of a picture by thresholding the area of every pixel and gets the outcome as a paired number. Because of its discriminative force and computational effortlessness, LBP has become a very well-known methodology in different applications. It very well may be viewed as a bringing together way to deal with the generally different factual and basic models of surface investigation. Maybe the most significant property of the LBP processing in true applications is its strength to monotonic dim scale changes caused, for instance, by enlightenment varieties. Another significant property is its computational effortlessness, which makes it conceivable to investigate pictures in testing continuous settings.

4.2.1 Modules and Software Used:

- Drone Flight Control and Image Capturing System.
- Python 3
- Image processing Pattern Techniques.

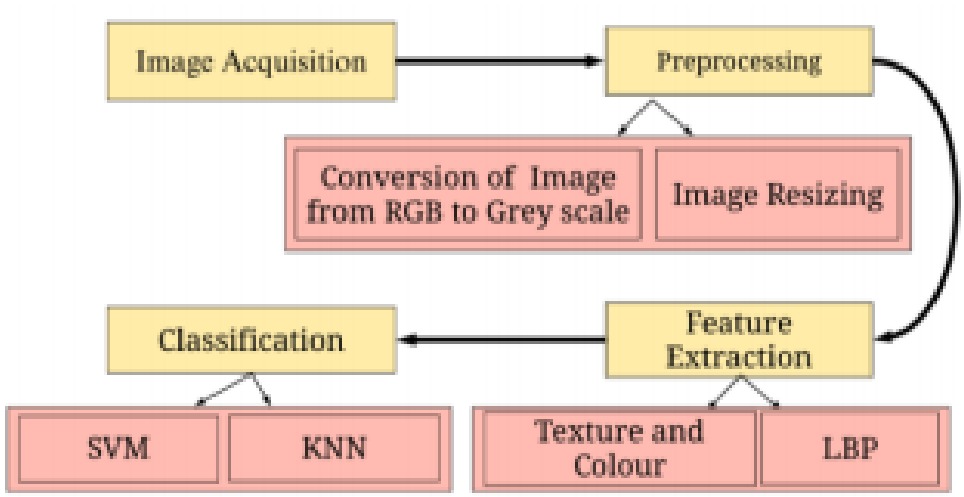


Fig 4.1 Block diagram of Disease detection system

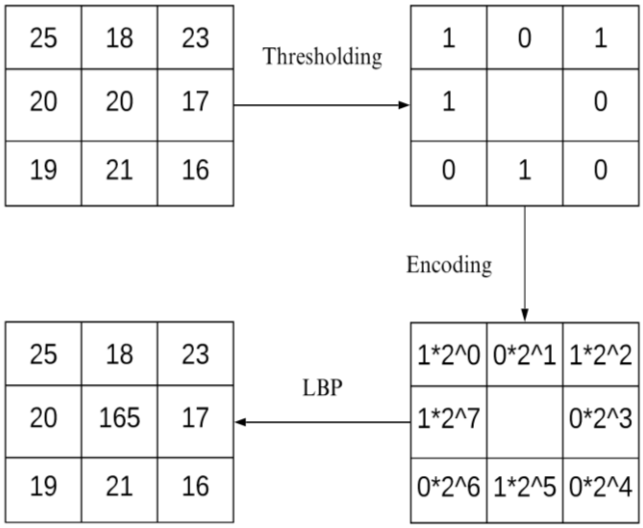


Fig 4.2 LOCAL BINARY PATTERN OPERATION

After preprocessing and feature extraction of leaf image, they are classified as healthy or diseased image using classifiers. Two experimental cases are considered in this work:

(i) Healthy-Black Sigatoka

(ii) Healthy-Cordana leaf spot.

Classification of images comprises of two major steps: training and testing processes. In this work, k-Fold cross-validation method is used to obtain the training and the testing datasets. To analyze the performance of the proposed algorithms CNN classifiers with 7-fold cross validation is used in this work. The statistical parameters like accuracy, sensitivity and specificity are used to examine the performance of the classifiers.

(a) (b) (c)

Fig 4.3 Sample picture of

(a) healthy banana leaf

(b) Black Sigatoka disease

(c) Cordana leaf spot

4.3 MATERIALS AND METHODS

The complete design of the drone was calculated by considering the total weight of the drone mounted sprayer as reference and these consideration parameters are payload capacity, design of supporting frame, fluid tank, selection motors, battery, propellers, flight controller, transmitter and receiver along with the Raspberry pie module.

4.3.1 METHODS and MATERIALS IN SOFTWARE

4.3.1.1 Dataset. Appropriate datasets are required at all stages of object recognition research, starting from training phase to evaluating the performance of recognition algorithms. All the images collected for the dataset were downloaded from the Internet, searched by disease and plant name on various sources in different languages. Images in the dataset were grouped into classes. In order to distinguish healthy leaves from diseased ones, one more class was added in the dataset. It contains only images of healthy leaves.

4.3.1.2 Image Pre-processing and Labelling. Images downloaded from the Internet were in various formats along with different resolutions and quality. In order to get better feature extraction, final images intended to be used as dataset for deep neural network classifier were pre-processed in order to gain consistency. Furthermore, procedure of image pre-processing involved cropping of all the images manually, making the square around the leaves, in order to highlight the

region of interest (plant leaves). During the phase of collecting the images for the dataset, images with smaller resolution and dimension less than 500 px were not considered as valid images for the dataset. In addition, only the images where the region of interest was in higher resolution were marked as eligible images for the dataset. In that way, it was ensured that images contain all the needed information for feature learning. Images used for the dataset were image resized to 256 × 256 to reduce the time of training, which was automatically computed by written script in Python, using the OpenCV framework.

4.3.1.3 Classification of the Images. Images that are being trained to the model are classified using Convolutional Neural Networks. The input image filters out the diseased or not diseased plants by binary classification.

4.4 TEST CASES AND RESULTS

The dataset which is trained using CNN into the Deep Learning model has given the following accuracy

```
Using TensorFlow backend.

Found 45 images belonging to 1 classes.
Found 45 images belonging to 1 classes.
Epoch 1/6
1000/1000 [==============================] - 228s 228ms/step - loss: 7.9844e-04 - accuracy: 0.9996 - val_loss: 4.6611
e-24 - val_accuracy: 1.0000
Epoch 2/6
1000/1000 [==============================] - 231s 231ms/step - loss: 1.7263e-16 - accuracy: 1.0000 - val_loss: 2.4916
e-22 - val_accuracy: 1.0000
Epoch 3/6
1000/1000 [==============================] - 231s 231ms/step - loss: 1.1953e-16 - accuracy: 1.0000 - val_loss: 9.0597
e-27 - val_accuracy: 1.0000
Epoch 4/6
1000/1000 [==============================] - 229s 229ms/step - loss: 1.4405e-16 - accuracy: 1.0000 - val_loss: 2.6187
e-25 - val_accuracy: 1.0000
Epoch 5/6
1000/1000 [==============================] - 237s 237ms/step - loss: 9.4761e-17 - accuracy: 1.0000 - val_loss: 3.1251
e-27 - val_accuracy: 1.0000
Epoch 6/6
1000/1000 [==============================] - 232s 232ms/step - loss: 1.3184e-16 - accuracy: 1.0000 - val_loss: 2.4911
e-22 - val_accuracy: 1.0000
Classifier trained Successfully!
```

Fig 4.4 Accuracy of the trained dataset

The test cases are checked with the healthy and unhealthy images which is a dataset of 100 images and the output is shown below

```
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
sigatoka
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
healthy
```

Fig 4.5 Prediction of 50images healthy and unhealthy

The test cases with the healthy and unhealthy images can be shown as below

```
if(result[0][0] == 1):
    print("I guess this must be healthy!")
else:
    print("I guess this must be a black sigatoka!")
    count+=1
```

```
Loaded model from disk
I guess this must be a black sigatoka!
```

Fig 4.6 black sigatoka image tested

```
    print("I guess this must be healthy!")
else:
    print("I guess this must be a black sigatoka!")
    count+=1
```

```
Loaded model from disk
I guess this must be healthy!
```

Fig 4.7 Accuracy of the trained dataset

4.5 RESEARCH OBJECTIVES

- To establish system for plant disease detection including causes and interventions.
- To investigate the challenges associated with plant disease and response service.
- To review existing techniques applied on the detection of plant disease and review their challenge.
- To develop a prototype that incorporates multiple pattern techniques to detect agricultural disease outbreaks

CHAPTER 5

Conclusion

It is concluded that a device is developed that extracts the features from pictures and classifies the ailments with the usage of deep learning Algorithms. This system is to develop the Drone module for real-time Disease identification with the help of Image processing modules. Disease identification with Image processing helps in identification of Disease on the spot while spraying of pesticide this helps in the reduction in the use of fertilizer and more man power in removal of Diseases. Fixed cameras are directly connected to a laptop for spot correction of images like cropping, colour correction, rotation, etc. after the correction the images are saved in a sequence which is captured based on lighting conditions that leads to the next step for creating datasets for all the images for further processes. The method that is followed to capture the images in farmland is fixing camera on the drone and also manual methods are used to collect images accurately without any texture and shading defects.

FUTURE ENHANCEMENTS

After analysing the test results of the developed systems, the following issues are open which can be taken up as future enhancements. As the drones have lesser standby time we need to improve the battery system that can work for longer period of time. In this project we have worked on banana plant but the classification and the pattern techniques can be used on other plant datasets to find the unhealthy plants. These applications can be used in other domains to carry out plant disease management efficiently.

APPENDIX A (SOURCE CODE)

LOCAL BINARY PATTERN:

```
import cv2 as cv
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from PIL import Image

#reading the picture
pic = cv.imread("IC420.jpg")

#displaying the picture
cv.imshow( 'original' , pic )

#converting the grayscle

pic = cv.cvtColor( pic , cv.COLOR_BGR2GRAY )

#displaying the grayscale picture

#cv.imshow( 'gray_scale' , pic )

cv.imshow( "pre-processed picture" , pic )
print(pic)



def get_pixel(img, center, x, y):
    new_value = 0
    try:
        if img[x][y] >= center:
            new_value = 1
    except:
        pass
```

```
    return new_value



def LBP(pic,h,w):
    img = pic

    center = img[w][h]
    val_ar = []
    val_ar.append(get_pixel(img, center, w-1, h+1))     # top_right
    val_ar.append(get_pixel(img, center, w, h+1))       # right
    val_ar.append(get_pixel(img, center, w+1, h+1))     # bottom_right
    val_ar.append(get_pixel(img, center, w+1, h))       # bottom
    val_ar.append(get_pixel(img, center, w+1, h-1))     # bottom_left
    val_ar.append(get_pixel(img, center, w, h-1))       # left
    val_ar.append(get_pixel(img, center, w-1, h-1))     # top_left
    val_ar.append(get_pixel(img, center, w-1, h))       # top

    power_val = [1, 2, 4, 8, 16, 32, 64, 128]
    val = 0
    for k in range(len(val_ar)):
        val += val_ar[k] * power_val[k]
    return val
shape= np.shape(pic)
#print(shape)
height =shape[1]
width = shape[0]
img_lbp = np.zeros((height, width,3), np.uint8)
for i in range(0, height):
        for j in range(0, width):
            img_lbp[i, j] = LBP(pic, i, j)
im=Image.fromarray(img_lbp)
im.save('lbp.jpg')
cv.imshow("lbp",img_lbp)
#rotating
```

```
lbp = Image.open("lbp.jpg")
rotated_90 = lbp.transpose(Image.FLIP_LEFT_RIGHT)
rotated_90 = rotated_90.transpose(Image.ROTATE_90)
print(np.shape(rotated_90))
rotated_90.show()
```

CLASSIFIER AND TESTER:

```
#!/usr/bin/env python
# coding: utf-8

# In[4]:


# Step 1: Import the packages

from keras.models import model_from_json

import cv2

import numpy as np


# Step 2: Load the Model from Json File

json_file = open('../../Classification/Image_classification_summary/model.json', 'r')

loaded_model_json = json_file.read()

json_file.close()

loaded_model = model_from_json(loaded_model_json)


# Step 3: Load the weights

loaded_model.load_weights("../../Classification/Image_classification_summary/model.h5")

print("Loaded model from disk")


# Step 4: Compile the model

loaded_model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

```
# Step 5: load the image you want to test
image = cv2.imread('C:/Users/Dell/Desktop/plant database/black sigatoka/')
image = cv2.resize(image, (50,50))
image = image.reshape(1, 50, 50, 3)

# cv2.imshow("Input Image", image)
# cv2.waitKey(0)
# cv2.destroyAllWindows()
# Step 6: Predict to which class your input image has been classified
result = loaded_model.predict_classes(image)
if(result[0][0] == 1):
    print("This is a plant effected with Black Sigatoka")
else:
    print("This is a healthy plant")

# In[5]:

for i in path:
    image=cv2.imread(i)
    image=cv2.resize(image,(50,50))
    image=image.reshape(1,50,50,3)
    result=loaded_model.predict_classes(image)
    print(Sigatoka)

# In[ ]:

import os
```

```
pathi="/Users/Dell /Desktop/plant Disease /Datasets/Sigatoka/test_set/Sigatoka /"

path_temp=os.listdir(pathi)

path_temp.sort()

path_temp.remove('_DS_Store')

# print(path_temp)

path=[]

for i in path_temp:

    path.append(pathi+i)

print(path)
```

TRAINER:

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from PIL import Image
# Step 2: Initialising the CNN
model = Sequential()

# Step 3: Convolution
model.add(Conv2D(32, (3, 3), input_shape = (50, 50, 3), activation = 'relu'))

# Step 4: Pooling
model.add(MaxPooling2D(pool_size = (2, 2)))

# Step 5: Second convolutional layer
model.add(Conv2D(32, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

# Step 6: Flattening
model.add(Flatten())

# Step 7: Full connection
```

```
model.add(Dense(units = 128, activation = 'relu'))
model.add(Dense(units = 1, activation = 'sigmoid'))

# Step 8: Compiling the CNN
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# Step 9: ImageDataGenerator
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen=ImageDataGenerator(rescale=1./255)
# Step 10: Load the training Set
training_set = train_datagen.flow_from_directory('C:/Users/Dell/Desktop/plant database/train
set/black sigatoka/',
                                                 target_size = (50, 50),
                                                 batch_size = 32,
                                                 class_mode = 'binary')
test_set=test_datagen.flow_from_directory('C:/Users/Dell/Desktop/plant database/test
set/',target_size=(50,50),
                                  batch_size=32,class_mode='binary')
# Step 11: Classifier Training
model.fit_generator(training_set,
                    steps_per_epoch = 1000,
                    epochs = 5,
                    validation_data=test_set,
                 validation_steps=100)

# Step 12: Convert the Model to json
model_json = model.to_json()
with open("../../Classification/Image_classification_summary/model.json","w") as json_file:
  json_file.write(model_json)

# Step 13: Save the weights in a seperate file
model.save_weights("../../Classification/Image_classification_summary/model.h5")

print("Classifier trained Successfully!")
```

APPENDIX B (OUTPUT SCREENSHOTS)

OUTPUT OF THE PREDICTION:

When tested with a black sigatoka image:

```
if(result[0][0] == 1):
    print("I guess this must be healthy!")
else:
    print("I guess this must be a black sigatoka!")
    count+=1
```

```
Loaded model from disk
I guess this must be a black sigatoka!
```

When tested with a healthy Image:

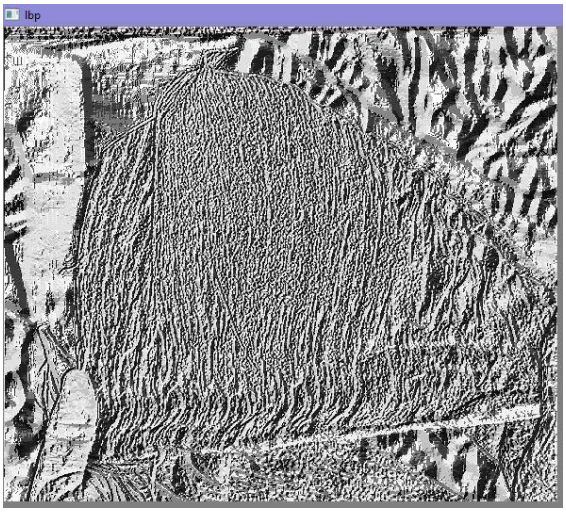
```
    print("I guess this must be healthy!")
else:
    print("I guess this must be a black sigatoka!")
    count+=1
```

```
Loaded model from disk
I guess this must be healthy!
```

OUTPUT OF THE PATTERN TECHNIQUE:

INPUT IMAGE

LOCAL BINARY PATTERN



References

- An adaptive approach for UAV-based pesticide spraying in dynamic environments [Bruno S. Faiçal etal., 2017]
- A survey of unmanned aerial sensing solutions in precision agriculture [Anandarup Mukherjee etal., 2017]
- Drone Agriculture Imagery System for Radish Wilt Disease Identification via Efficient Convolutional Neural Network [L. Minh Dang etal., 2018]
- An IoT based smart irrigation management system using Machine learning and open source technologies [Amarendra Goap etal.,2017].
- IoT and agriculture data analysis for smart farm [Jirapond Muangprathub etal., 2018]
- Develop an unmanned aerial vehicle based automatic aerial spraying system [Xinyu Xue etal., 2016].
- Deep learning models for plant disease detection and diagnosis[Konstantinos P. Ferentinos etal.,2018]
- Deep learning in agriculture: A survey[Andreas Kamilaris etal., 2018]
- www.sciencedirect.com
- www.ebscohost.com
- www.realpython.com
- www.ieeexplore.ieee.org
- www.scielo.org.co